Towards a Computational Model of Musical Accompaniment: Disambiguation of Musical Analyses by Reference to Performance Data

Benjamin David Curry

Doctor of Philosophy

Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh

2002

Abstract

A goal of Artificial Intelligence is to develop computational models of what would be considered intelligent behaviour in a human. One such task is that of musical performance. This research specifically focuses on aspects of performance related to the performance of musical duets.

We present the research in the context of developing a cooperative performance system that would be capable of performing a piece of music expressively alongside a human musician. In particular, we concentrate on the relationship between musical structure and performance with the aim of creating a structural interpretation of a piece of music by analysing features of the score and performance.

We provide a new implementation of Lerdahl and Jackendoff's Grouping Structure analysis which makes use of feature-category weighting factors. The multiple structures that result from this analysis are represented using a new technique for representing hierarchical structures. The representation supports a refinement process which allows the structures to be disambiguated at a later stage.

We also present a novel analysis technique, based on the principle of phrase-final lengthening, to identify structural features from performance data. These structural features are used to select from the multiple possible musical structures the structure that corresponds most closely to the analysed performance.

The three main contributions of this research are:

- An implementation of Lerdahl and Jackendoff's Grouping Structure which includes feature-category weighting factors;
- A method of storing a set of ambiguous hierarchical structures which supports gradual improvements in specificity;
- An analysis technique which, when applied to a musical performance, succeeds in providing information to aid the disambiguation of the final musical structure.

The results indicate that the approach has promise and with the incorporation of further refinements could lead to a computer-based system that could aid both musical performers and those interested in the art of musical performance.

Acknowledgements

First and foremost I wish to thank my supervisors Geraint A. Wiggins and Gillian Hayes who have provided an amazing amount of support and encouragement during the period of this research.

I would also like to thank my colleagues at Xilinx who have offered invaluable support over the past couple of years - especially Jane Hesketh and Scott Leishman who have offered continuous encouragement and wisdom.

The EPSRC kindly funded me for three years under UK EPSRC postgraduate studentship 97305827. This allowed me the freedom to explore, learn and create in the wonderfully informal atmosphere of the Department of Artificial Intelligence at the University of Edinburgh.

Although performing research and writing a thesis is generally a solitary task, a number of people have been there to offer friendly advice, thoughtful conversations, words of encouragement, practical support, proof-reading and/or excuses to go for a drink. The following people belong to one or more of these categories and I will always be indebted to them: Angela Boyd, John Berry, Márcio Brandão, Neil Brown, Colin Cameron, Simon Colton, Stephen Cresswell, Jacques Fleuriot, Jeremy Gow, Kathy Humphry, Nathan Lindop, Ruli Manurung, Luke Phillips, Somnuk Phon-Amnuaisuk, Kaska Porayska-Pomsta, Thomas Segler, Joshua Singer, Craig Strachan, Gordon Reid, Angel de Vicente and the AI-Ed and AI-Music groups.

Finally I wish to thank my examiners Alan Smaill and Gerhard Widmer whose feedback has greatly improved this final thesis.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Benjamin David Curry*)

Publications

Some material in this thesis has already been published in the following sources (copies of which are included in Appendix D):

Ben Curry and Geraint A. Wiggins. A new approach to cooperative performance: A preliminary experiment. *International Journal of Computing Anticipatory Systems*, 4:163–178, 1999.

Ben Curry, Geraint A. Wiggins, and Gillian Hayes. Representing trees with constraints. In J. Lloyd *et al.*, editors, *Proceedings of the First International Conference on Computational Logic*, volume 1861 of *LNAI*, pages 315–325. Springer Verlag, 2000.

To Michael, Carol and Jacob.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

One of the aims of the artificial intelligence community is to create computational systems that can perform tasks which are considered to require intelligence in humans. The skill of musical performance is one such task.

When a human musician performs a piece of music, they do not follow the musical score exactly, but instead use their knowledge about the piece of music to manipulate the performance to emphasise certain aspects of the piece being performed. The act of manipulating the performance of the notated musical events is called *expressive performance.*[1]

This thesis explores the hypothesis that there is a link between *musical structure* and expressive performance and, that by exploiting this link, a computer-based system can be created that is capable of performing a piece of music expressively alongside a human musician in a duet context.

1.1 The Problem

Systems to accompany[2] human musicians have been developed previously (e.g. Dannenberg and Mukaino (1988); Raphael (2001)) which use score-tracking techniques

[1] Terms which may be unfamiliar to some readers are highlighted by *italics* the first time they occur. Their definition will be provided either within the surrounding text or in the glossary at the end of this thesis.

[2] The terms duet and accompaniment are used interchangeably within this thesis to describe two musicians collaboratively performing a piece of music.

to follow the current performance and adapt accordingly. For example, if a musician varies tempo or dynamics during the performance, the system will similarly vary its own performance to match the human musician. The weakness of these systems is that they are adopting a passive, or reactive, rôle during the performance. Specifically, the systems are designed to base their performance solely on the current performance and react to the musical events performed by the human musician. They have no expressiveness other than that derived from the human.

There are both practical and musical problems with this passive approach. From a practical perspective; how should the system behave during long periods of solo performance when the performance of the other musician offers no guidelines? How should the system react when a badly-timed event is performed?

From a musical perspective; the performance of a duet requires the cooperation of two performers which leads to a shared model of the musical structure of the piece (Appleton et al., 1997). How can a passive system contribute to such a model?

To remedy these weaknesses, a computer-based accompaniment system is proposed that infers knowledge of the musical structure of the piece being performed. The system will adopt an active rôle during the performance by using the inferred musical structure as a guide for generating an expressive performance. The system will still adapt to the human musician, but it will no longer be entirely subservient.

Achieving this poses further questions. The musical structure of a piece is open to interpretation by each musician that performs it. For example, a certain piece may have a musical structure that supports either a two-bar or four-bar phrase structure which are both equally valid. If this is the case, then the two musicians performing the piece will have to agree on the musical structure (i.e. have a shared model) to avoid conflicting expressive gestures.

This implies that, in order for the proposed system to be an effective accompanist, it will have to incorporate a model of the musical structure which is shared with the human musician.

It has been shown that musical structure influences how a musical piece is performed. The novel approach presented in this thesis is to invert this relationship in order to derive the musical structure, i.e. take an expressive musical performance and

from that analyse the structure of the piece. This final step is the main focus of the research presented in this dissertation.

The following section describes how the results of this research may lead to a useful tool for musicians and researchers.

1.2 Applications of the Research

If the approach described above is successful, and can be included into a computer-based accompanist, the resulting system could be used to provide greater insight into aspects of performance and would be useful for:

- Education - allowing students of musical performance the ability to practise a piece of music without the need for a musical partner;
- Performance - by enabling a musician to experiment with different interpretations of a piece and see how that alters the accompanying performance;
- Research - if the system is sufficiently modular, researchers could investigate the results of applying different theories of musical structure or performance within the system.

The next section presents the aims of the research presented in this dissertation.

1.3 Aim of the Research

The main aim of this research is to investigate whether it is possible to create a computational system that can generate a model of the musical structure of a piece which is informed by both the expressive performance of that piece and the piece's musical score.

This principal aim can be divided into a number of smaller goals:

1. Provide a rule based implementation of a proven theory of musical structure that supports multiple structures and then subsequent refinement;

2. Develop a technique for analysing musical performance in order to provide information about the musical structure of a piece;

3. Produce a structural interpretation of a piece based upon its performance using results from the above two goals.

1.4 Achievements

This dissertation contains solutions to the above goals. The first goal is met by providing an implementation of Lerdahl and Jackendoff's (1983) Grouping Structure (see Chapter 4) that supports a gradual refinement process.

The gradual refinement process allows the initial grouping analysis to contain many more grouping boundaries than will actually be present in the final analysis. These extraneous boundaries will be removed by making use of information derived from the piece's performance.

This dissertation presents both a new implementation of the Grouping Structure which incorporates feature-category weighting factors (see Chapter 4) and, separately, a generic and novel tree-based representation which supports gradual refinement (Chapter 5).

The second goal is solved by the performance analysis module which takes as input a database of previous performances and, from these, identifies potential musical features. In order to show that expressive performances remain mostly consistent across a period of time, and to gather data for the performance analysis, an empirical study was performed (Chapter 6).

The analysis of the musical performances is based on the concept of phrase-final lengthening. The performance analysis process searches the musical performance for repeating occurrences of convex curves in the timing data. If the musical structure of the piece has some form of regularity, this regularity should manifest itself as a series of convex curves. Instances of these curves provide clues which aid the identification of phrase boundaries in the performance. A novel means of detecting these phrase boundaries is described and the application of this technique to three different musical pieces is presented (see Chapter 7).

The final goal is achieved by incorporating the results from the grouping structure analysis and the performance analysis. The musical structure for three musical pieces is derived by the synthesis of the above analyses. The resulting musical structures are subsequently evaluated and show that the synthesis of the performance and structural analyses does contribute towards selecting a valid musical structure for the analysed pieces (Chapter 9).

1.5 Thesis Structure

The structure of the thesis is as follows:

Chapter 1: Introduction introduces the problem to be solved and the related questions.

Chapter 2: Related Work presents a survey of research related to the issues addressed by this thesis.

Chapter 3: System Overview provides an overview of the structure of a system designed to perform expressively alongside a human musician.

Chapter 4: Structural Analysis describes how a partial implementation of Lerdahl and Jackendoff's grouping structure which includes feature-category weighting factors was developed.

Chapter 5: Tree Representation presents a novel representation for tree structures using constraint logic programming which is used to represent the results of structural analysis.

Chapter 6: Empirical Study describes an experiment to gather performance data and show the consistency that exists between different performances of the same piece.

Chapter 7: Performance Analysis presents two different analysis techniques designed to identify repeating timing structures. These are applied to the data gathered in the study with the aim of identifying structurally significant parts of the music.

Chapter 8: Feature Detection describes how the results from the performance analysis process can be analysed to detect important features.

Chapter 9: Synthesis demonstrates how the information from the feature detection process and the structural analysis can be combined to create a structural representation corresponding to the way the musicians performed the piece.

Chapter 10: Conclusions and Further Work gives the conclusions that can be drawn from this work and highlights issues worthy of further investigation.

Appendix A: Charm Representation presents the musical events of the pieces used in this research encoded using the *Charm* representation (Smaill et al., 1993) discussed in Chapter 4.

Appendix B: Grouping Analyses contains the results of applying the grouping analysis module to the Charm representations of the musical pieces.

Appendix C: Partial Autocorrelation presents the mathematical details of the partial autocorrelation technique used in Chapter 7.

Appendix D: Published Papers includes published papers which have presented some of the work contained in this thesis.

Appendix E: Musical Scores contains the scores of the three musical pieces used in Chapters 4–9.

Chapter 2

Related Work

This chapter contains an overview of work related to this research. A wide range of topics from expressive performance to musical structure are discussed.

2.1 Introduction

The chapter begins with a description of what constitutes an expressive performance and the way musicians create them. It then presents research that suggests that expressive performance can be modelled artificially and that performance relies significantly upon the musical structure of a piece. Three theories of musical structure are introduced, and parallels are drawn between their similar aspects. The chapter describes some research related to real-time performance tracking and duet performance, and closes with some observations about the existing research.

2.2 Expressive Performance

A *musical score* acts as a guide to a musical performer; it does not prescribe exactly how a piece is to be performed. The musician uses their own skills and intuitions to perform the piece with altered features, such as changes in *timing* or *dynamics*, in order to create an expressive performance. A performance which is not expressive is typically called a *mechanical performance*.

An expressive performance enhances a piece of music by emphasising certain aspects of the music which the musician feels are important to convey to the listener.

Canazza et al. (1997) and De Poli et al. (1998) performed experiments to determine what global[1] aspects of a performance were manipulated by performers to convey different emotional states. The performers were asked to perform pieces in a style appropriate to particular keywords such as light, heavy, bright and dark. The keywords were chosen to be non-standard musical words.

Analysis of the results showed that the performances could be separated along two different principal axes, one representing brightness and the other softness. It was discovered that these two axes correspond to the *tempo* of the piece and the *amplitude envelope* of the notes respectively.

A performance can also be manipulated at the local level by performers altering the timing of the individual musical events. Repp (1995) defines the *timing micro-structure* as the "continuous modulation of the local tempo, resulting in unequal intervals between successive tone onsets, even if the corresponding notes have the same value in the score". In preliminary experiments investigating whether global tempo and timing micro-structure are independent or not, Repp (1994b) found two contradictory results described below.

When two pianists were asked to perform a piece of music at slow, medium and fast tempi, it was found that the most expressive performance was the one performed at medium tempo.[2] In both of the other performances, the musicians decreased the amount of deviation introduced into the performance. Repp had intuitively expected that the performance at the fast tempo would provide the least expressive performance, and the slow performance would have the greatest amount of expressive deviation.

To further investigate this issue, Repp performed another experiment asking listeners to judge a number of performances for aesthetic quality. The performances presented to the listeners consisted of fifteen examples in total, five examples for each of three global tempo speeds with each of the five examples varying in the size of expres-

[1] A distinction is drawn between 'global' aspects of performance that remain relatively consistent throughout the performance and 'local' aspects which are continuously altered throughout the piece.

[2] Medium tempo corresponded to the musician's 'natural' tempo for that piece; the slow and fast speeds were approximately 15% slower and faster than this natural tempo.

sive deviations. The results showed that the listeners' judgement policy agreed with Repp's earlier intuition: for the fast performances, the listeners preferred a decrease in *expressive deviation* and, with slightly less significance, they preferred an increase in deviations for the slow performances.

Repp speculates that the different results of these two experiments might be due to the musical performers in the first experiment being slightly uncomfortable at performing pieces at a unusual tempo which in turn restricted the amount of expression that they were able to use. He suggests that if the musicians had been given more time to practice and get used to the piece at the different tempi, more expression would be introduced.

Desain and Honing (1994, 1992) performed similar investigations into the relationship between global tempo and the timing micro-structure. They analysed how the timing profile changed as the tempo of a piece was altered. The results showed that "[local expressive] timing, in general, does not scale proportionally with respect to global tempo" (Desain and Honing, 1994 p.16).

Another source for musical expression, apart from timing and dynamics, is the *timbre* of the events being performed. Different instruments offer the performer different ways of manipulating the events being performed. Because many timbral aspects of performance are instrument specific, this research will concentrate on methods of adding expressivity that are relatively instrument independent such as timing and dynamics.

The research presented above leads to the conclusions that there are two distinct forms of expression: one which occurs at a global level (e.g. the overall tempo of a piece) and the other that is more local in nature (e.g. the timing fluctuations within a phrase). Although the two are related, a change in one will not necessarily induce a change of similar proportion in the other.

2.2.1 Consistency

In order to model the process of musical expression it is necessary to investigate the properties of an expressive performance. The most significant aspect for this research is that of consistency. If, for a given piece of music, there is a 'standard' expressive

performance then there is the possibility of a transformation from musical score to expressive performance that could be modelled by a computer system. The challenge will then be to find this transformation.

Repp (1997b) performed a comparative study of two different groups of pianists performing the same piece. One group consisted of ten recorded performances by professional musicians and the other of ten graduate students whose performances were recorded using MIDI. Repp found that the "*average* expressive timing profiles were extremely similar" (Repp, 1997b p.257).

Individual performances tended to be more different amongst the group of expert musicians than among the students.[3] Despite these differences, the commonalities of the performances suggests that there is a "common standard of expressive timing" (*ibid.*, p.257). In Repp (1995, 1997b), comparative studies were performed which came to similar conclusions about the similarity of the average timing profiles of two groups of experts and students.

Assuming that the expert pianists had performed many rehearsals before making the final recording and that the student pianists had little experience of the piece, Repp concludes that the common standard "may be considered the default result of an encounter between a trained musician and a particular musical structure - the timing implied by the score" (Repp, 1997b p.257).

After discovering how similar different musicians' performances of the same piece could be, Repp investigated how listeners would judge a performance created from the statistical average of a number of expressive performances (Repp, 1997a). In one experiment, where listeners were asked to rate eleven performances, one of which was the average performance, the listeners judged the average performance to be second highest in quality and second lowest in individuality. A second experiment used thirty performances by professional musicians and student musicians with three average performances included, one the average of the professional musicians, another the average of the student musicians and the last the average over all the performances. The listeners rated all of the average performances highly, and found the average expert performance to be the best of the thirty examples.

[3] Repp tentatively speculates that this may be due to the pressure that a professional artist is under to be different from their peers.

There are three interesting results to Repp's research:

- There is a high amount of consistency between performances of the same piece by the same musician;
- There is a high amount of similarity between performances of the same piece by different musicians;
- An average performance can be considered to be of high quality.

These results suggest that there is an established way of performing a piece of music expressively and that the expression is relatively invariant between performers and performances. The fact that a performance created from the average performance of a number of musicians is considered to be high quality suggests that there is an underlying timing structure common to all of the performances which is conveyed through the average performance.

2.2.2 Modelling

This section presents research that investigates and attempts to model various aspects of expressive performance.

Arcos et al. (1997) use case based reasoning to generate Jazz style expressive saxophone performances. Their system, SaxEx, works by storing a set of scores and associated expressive and inexpressive performances and uses these to generate an expressive performance of a new piece. The new piece is analysed note by note by the SaxEx system which searches for similar cases in its case base. If it finds any similar cases, i.e. notes within a similar structure, it preferentially ranks the matches and then applies a transformation based upon the best match to the inexpressive performance of the new piece. Although there has been little reported experimental evaluation of the system, audio samples of the output of SaxEx does demonstrate the system adding colour to what was originally an inexpressive performance.

Juslin (1997) performed two different experiments investigating listeners' judgements of emotional intent in synthesised performances of a short melody. The first

investigated the musical cues involved in conveying five different emotional expressions: happiness, sadness, anger, fear and tenderness. The cues that were manipulated included tempo, sound level, timing, *attack* and *vibrato*. A set of listeners were asked to judge a mix of synthesised and 'real' performances for their emotional content. The results showed that the listeners were successful in identifying the intended emotion and that the proportion of these correct judgements were equivalent for both the synthesised and the real performances. The listeners were also presented with the same set of performances played backwards. For these reversed performances, it was found that listeners had more difficulty decoding the expressive intention of the real performances than for the synthesised performances. Juslin felt that this suggested that the real performances "were relatively more dependent on prosodic contours" Juslin (1997 p.225).

The second experiment looked at the various contributions made by five of the cues. The cues that were analysed in this experiment were: tempo, sound level, frequency spectrum (one of soft, bright or sharp), *articulation* and attack. The listeners were asked to judge 108[4] different performances and rate them on six adjective scales: angry, happy, sad, fearful, tender and expressive. The experiment showed that all of the cues played an equal part in dictating the listeners' judgements.

Canazza and Rodà (1999) and Rodà and Canazza (1999) describe a system that can add expression to a musical performance in real time. The model is based on the results of their experiments mentioned above (Section 2.2, p.8) that investigated what aspects of performance were manipulated in order to convey emotions. The system calculates how to manipulate the acoustic properties of a performance based upon a user input of the desired expressive quality of the performance. The expressive quality can be manipulated in real time so a piece can begin as quite 'heavy' and then gradually change to be 'bright' as the performance progresses. The specification of the expressive quality is done by using a two dimensional space that represents how listeners had segmented pieces of music performed with different emotional intent. As the expressive quality is altered, the system manipulates several acoustic parameters such

[4] All possible cue combinations: tempo (slow, medium, fast), sound level (low, medium, high), frequency micro-pauses spectrum (soft, bright, sharp), articulation (*legato*, *staccato*) and attack (slow, fast).

as tempo, intensity or attack time. The real-time demonstration of their model does convey a sense of the intended emotion but without an understanding of the influence of the structure of the piece on the performance, the system is quite limited.

To investigate the plausibility of a generative model that transforms a musical structure into an expressive performance, Clarke (1993) performed an experiment testing the ability of performers to imitate another performance. The given performances were either examples of 'real' performances or artificially generated ones that would disrupt the relationship between the structure and performance. The results showed that the more disrupted the relationship was, the harder it was for the musicians to accurately repeat the performance.

In a similar experiment, Repp (1992b) examined the interaction between musical structure and timing. The experiment consisted of asking listeners to identify the locations of randomly inserted micro-pauses in the performance of an eight-bar musical excerpt. The listeners found it hardest to identify these pauses when they occurred where lengthening would typically arise in an expressive performance of that phrase. This implied that the listeners had expectations about when lengthenings would occur during the performance of a piece and that the locations of these lengthenings corresponded with the structural features of the piece.

A set of experiments by Amandine Penel and Carolyn Drake (1997) examined how lower-level features at the musical surface affect performance. They found that for a series of tones, the interval between the last two notes was consistently longer than the preceeding interval by between 10% and 20%. Examining the timing of low-level phrases, Penel and Drake found that "each grouping process seems to result, in musical performance, in a ritardando or accelerando/ritardando profile" (Penel and Drake, 1997 p.470).[5]

Friberg and Bresin (1997) and Friberg et al. (1997) present a system for automatic musical punctuation as part of a rule based system for musical performance. The long-term goal of the research is to "obtain a better understanding of the mechanisms underlying musicians' transformation of written scores into a musically convincing performance" (Friberg et al., 1997 p.719). The musical punctuation mentioned above

[5]The terms *ritardando* and *accelerando* are defined in the glossary.

is the insertion of low level boundaries into the score. These boundaries are used to alter the performance by introducing and tone lengthenings where the boundaries occur. The resulting punctuation was considered successful with an external judge considering that 90% of the generated punctuations were musically acceptable.

Todd (1989a,b) presents a computational model of *rubato* based upon Lerdahl and Jackendoff's Time-Span Reduction (see Section 2.3.3). The model is intended to produce an output that corresponds to the rubato that a performer might use based upon the *phrase structure* of the piece (see e.g. Repp (1992a)). Todd restricts the amount of knowledge his model has access to in order to more closely resemble the memory limitations of a human musician (an enhancement of Todd (1985)). In his model, the system represents the structure of the piece as a forest of trees, rather than one large connected tree, with one high level tree connecting the lower level trees. This representation encapsulates the idea that the performer has detailed access to local information as well as some high level overview of the piece but cannot necessarily compare any event with any other event.

Todd's model uses a parabolic function to represent the timing deviations that correspond to the phrasal structure. These timing alterations can be applied recursively at different heights of the structure to create a timing profile for the piece. The parabolic function has six parameters that tune the strength and shape of the curve to the current context by including parameters such as tempo, amplitude, length of phrase and boundary strength.

In Todd (1992), the model is complemented by another "based on the observation that a musical phrase is often indicated by a crescendo/decrescendo shape" Todd (1992 p.3540).[6] This proposed model of musical dynamics is closely related to his timing model as the dynamics of the performance are influenced by the tempo and the structural significance of the phrase boundaries.

Widmer uses machine learning to generate expressive performances from the score. In Widmer's earlier work (Widmer, 1995a,b), a system was developed that learnt how to perform a piece expressively by creating rules for generating expression based upon information at the note level of the score with an explicit user-created indication of

[6]The terms *crescendo* and *decrescendo* are defined in the glossary.

the structural rôle the note plays. The input to the system was a representation of the score of a piece, some higher-level structural information for each event and the tempo curves that represented the timing deviations introduced by the performer. From these inputs, the system generated a set of general rules that specify under what conditions a timing deviation should apply and what size of deviation it should be.

Widmer (1996, 1997) improves the earlier work by concentrating on the structural level of the pieces being performed. The system creates rules that link structural properties with expressive shapes for dynamics and rubato. Widmer states that this leads to "smoother and more balanced performances and is also more plausible from a musical point of view" (Widmer, 1996 p.179).

An approach to expressive performance proposed by Parncutt (1997) attempts to model some of the physical aspects of performance. He argues that the computer-based systems that generate expressive performances could be improved[7] by incorporating some of the physical properties, and limitations, of human musicians.

The research discussed above varies greatly in the techniques used to generate and analyse expressive performances, but one common theme occurs throughout most of the research. The research tends to use more than just information in the score, it includes, to varying degrees, some notion of the musical structure of the piece.

The studies investigating the link between musical structure and expressive performance clearly show that such a link exists and, when that link is deliberately disrupted, the results confuse the experimental subjects. Musical structure is an important part of expressive musical performance.

2.3 Musical Structure

This section discusses the concept of musical structure and what extra information musical structure provides over the score. It begins with a discussion of what a musical structure is and then gives an overview of some theories of musical structure and compares similar components of the presented theories.

The musical score suggests aspects of the musical events to be performed such

[7] Specifically, made to sound more 'genuine'.

as the *pitch*, duration, loudness and perhaps some timbral information. The score may also contain information relating to the phrasing of parts of the piece and *accents* applicable to individual musical events. However, it does not contain information about *motifs*, *themes*, *tension*, *harmonic progression* or other higher level aspects of the piece.

These higher level aspects of the piece are fundamental to an understanding of the piece and emerge from listening to the piece of music being performed. One of the assumptions of this thesis is that the musical score is an important source of information when trying to develop a structural understanding of the piece.

The following are examples of theories of musical structure that produce a musical structure from a representation of the musical surface of a piece.

The Generative Theory of Tonal Music (GTTM) is a rule based theory that attempts to model *"the musical intuitions of a listener who is experienced in a musical idiom."* (Lerdahl and Jackendoff, 1983 p.1)

The theory consists of four parts dealing with different aspects of musical structure. The specification of each part is made by a series of well-formedness, preference and transformational rules. The well-formedness rules specify what structures are possible. The preference rules select from the set of possible structures those that correspond most closely to the musical surface. Finally, the transformational rules allow the creation of specific exceptional structures that would disagree with the well-formedness rules.[8]

Narmour (1992) proposes a theory of musical structure based upon ideas of expectation and realisation. The Implication-Realisation Model (IRM) consists of two different types of 'process'. The top-down processes are idiom specific and are learned by the listener through exposure to music of a particular genre. The bottom-up processes are intended to be style-independent and based on the Gestalt principles of proximity, similarity and common fate. Krumhansl (1995) provides a good overview of the bottom-up analysis and includes some supportive experimental evidence.

Cambouropoulos (1998) proposes a General Cognitive Theory of Musical Structure (GCTMS) which "is a theory that may be employed to obtain a structural description (or set of descriptions) of a musical surface. [GCTMS] is independent of any

[8] See Lerdahl and Jackendoff (1983 pp.55-62) for a more detailed discussion and examples.

specific musical style or idiom, and can be applied to any musical surface" (*ibid.*, p.3).

The following sections introduce and discuss different aspects of musical structure which allow for a comparison across the three theories mentioned above.

2.3.1 Segmentation

When examining how a piece of music can be segmented into groups of events, there are two main points of interest; the boundaries that create the segmentation and the relationships between different groups.

Boundaries

Any group must have a start and end point: in the case of music these start and end points are usually indicated by a musical event or transition between events that is in some way distinguishable from the surrounding ones.[9]

GTTM's concept of grouping boundaries is based upon Gestalt principles of similarity and difference. The rules examine four adjacent notes at a time and examine if there is any change between the central two notes that is more distinct than the changes between the first two and the last two notes. These changes might be a change in duration, a greater time difference between the attack times or any other quantifiable change. Lerdahl and Jackendoff also suggest that other rules could be added to deal with less quantifiable aspects such as a change in timbre or instrumentation.

The grouping rules of GTTM have been largely supported through experimental testing. Deliège (1987) conducted experiments to see if listeners segmented music as predicted by GTTM. The results showed a broad agreement with the rules. The experiments were performed with both musically experienced subjects and non-musicians. It was found that the non-musicians were slightly less consistent with the grouping rules, but overall the two subject groups had similar grouping principles. Clarke also investigated the grouping rules and found that "listeners used segmentation criteria that were broadly consistent with the predictions of Lerdahl and Jackendoff" (Clarke and

[9] Interestingly, it has been shown that if you present a listener with a regular sequence of clicks, such as those produced by a metronome, the listener will tend to segment the clicks into groups of two or three (Miller (1962) from Deliège (1987)).

Krumhansl, 1990 p.248).

The Local Boundary Detection Model (LBDM) from GCTMS is also based upon Gestalt theories. LBDM consists of two rules: the Identity-Change rule and the Proximity rule. Unlike Lerdahl and Jackendoff's grouping rules, the Identity-Change rule does not identify a specific place to put a boundary, but identifies sequences of three objects that have some change occurring across them. It is then the job of the proximity rule to pick whichever of those boundaries is more pronounced and to highlight that as being more significant.

In the IRM, the grouping boundaries of a series of notes arise whenever 'closure' occurs. Closure is defined by Narmour to relate to "syntactic events whereby the termination, blunting, inhibiting, or weakening of melodic implication occurs." If a high degree of closure is encountered, then a structural transformation will occur which corresponds to a hierarchical grouping of the appropriate musical events. Narmour gives six conditions which can lead to the establishment of melodic closure (a combination of which implies a stronger sense of closure):

> 1. when simple stopping takes place, that is, when a rest, an onset of another structure, or a repetition interrupts an implied patterning;
> 2. when metric emphasis is strong;
> 3. when consonance resolves dissonance;
> 4. when duration moves cumulatively (short note to long note);
> 5. when intervallic motion moves from large interval to small interval; and
> 6. when registral direction changes (up to down, down to up, lateral to up, lateral to down, up to lateral, or down to lateral).
>
> Narmour (1992)

Parallelism

Another method of segmenting a stream of events into groups is to look for repeating patterns of events and then label each repetition as a group. In music, a theme or motif would qualify as such a pattern.

GTTM uses parallelism by stating that “where two or more segments of the music can be construed as parallel, they preferably form parallel parts of groups” (Lerdahl and Jackendoff, 1983 p.51). In other words, if two segments can be identified as being similar then their grouping structures should also be similar. However, Lerdahl and Jackendoff do not explicitly define what they mean by parallel: “we feel that our failure to flesh out the notion of parallelism is a serious gap in our attempt to formulate a fully explicit theory of musical understanding” (*ibid.*, p.53).

GCTMS has two different methods for identifying repeating patterns in a piece. The Sequential Pattern Matching Algorithm (SPMA) induces patterns by working bottom-up from the smallest possible patterns up to patterns of a maximum length. One of the major differences between this and GTTM’s parallelism is that it allows patterns to overlap, in fact patterns that do not overlap are the exception.

The second technique is the Unscramble algorithm which “given a set of objects and an initial set of properties, generates a range of plausible classifications for a given context” (Cambouropoulos, 1998 p.125). The Unscramble algorithm groups objects, in this case the musical segments discovered by LBDM, SPMA and the metrical structure (see below), and classifies them into categories according to their similarity.

IRM, in a similar way to GTTM, makes use of parallelism (or repetition) as an important source of structural information. However, no attempt is made to define how parallelism could be detected or measured.

2.3.2 Metrical Structure

The metrical structure of a piece assigns an alternating degree of importance to a series of events. For example, in simple quadruple time the first beat of the bar is often emphasised the most, the third beat is next strongest and then the second and fourth beat are equivalent in beat strength.

The metrical structure of GTTM is defined as “the regular, hierarchical pattern of beats to which the listener relates musical events” (Lerdahl and Jackendoff, 1983 p.17). Lerdahl and Jackendoff draw a distinction between three types of accent that are relevant to the metrical structure: *phenomenal accents* are events at the musical surface that “give(s) emphasis or stress to a moment in the musical flow” (*ibid.*, p.17), *struc-*

tural accents are implied by the melodic or harmonic structure of the piece and *metrical accents* are beats that are relatively strong given a metrical context. A metrical structure is then "a mental construct, inferred from but not identical to the patterns of accentuation at the musical surface" (*ibid.*, p.18).

The well-formedness rules specify a hierarchical structure of alternating weak and strong beats that begin from the *tactus* level of the piece.[10] The preference rules select a metrical structure that is consistent with the grouping structure and try to make strong beats lie on accented musical events.

The metrical structure of GCTMS is determined by trying to match a metric grid[11] to the *accentuation structure* (see Section 2.3.3) of the piece. The metrical structure is chosen by "adding the accents of all the events whose inception coincides with the points of the grid" (Cambouropoulos, 1998 p.107) and selecting the grid and position that creates the greatest score. If the same grid scores similar values at different offsets, then the piece is considered to be metrically ambiguous with respect to that grid.

In both these theories, the metrical structure relies heavily upon the grouping structure, and conversely, once a metrical structure is chosen it has a significant effect on the grouping structure.

2.3.3 Surface Reduction

Some passages of a piece of music can be considered to act as elaborations of others. The purpose of a surface reduction is to identify the core musical events of the piece being analysed.

GTTM's time-span reduction is based on their reduction hypothesis that the "listener attempts to organize all the pitch-events of a piece into a single coherent structure, such that they are heard in a hierarchy of relative importance" (Lerdahl and Jackendoff, 1983 p.106). The time-span reduction identifies the 'head' of each group proposed by the grouping structure. These heads are the most significant pitch event in that group. The preference rules identify the head based upon its tonality. If an event is relatively consonant or related to the local tonic then it will be selected as the head. The other

[10] Although a short-term lower level of beats may be introduced at certain points in the piece.

[11] A metric grid consists of a series of strong and weak beats that are regularly spaced in terms of the tactus level of the piece.

events in the group that are not as consonant as the head are seen as elaborations of the head. The heads of groups are similarly compared and the most stable lower level heads become the heads at the next level. The most stable head of all is the one that occurs at a cadence, with all other events being viewed as elaborations of that cadence.

IRM deals with surface reduction in a similar way. The most significant events in terms of the position within melodic structures are passed up through the levels of the analysis. At each level of the analysis, these events are again encompassed within structures and the defining events of these structures are considered the most significant.

In GCTMS, the accentuation structure performs only one level of surface reduction by identifying the accent strengths of the musical surface. The structure is derived from the LBDM by adding the boundary strengths of every pair of successive intervals. This gives an indication of importance at the musical surface but is not carried any further. There is no indication of higher level importance.

2.3.4 Tension/Relaxation

Another important aspect of music is the sense of tension and relaxation that is experienced whilst listening to a piece.

GTTM captures this notion with the use of prolongational reduction. The prolongational reduction captures a hierarchy of tensing and relaxing gestures in the music using a tree-based representation. The hierarchical nature of the representation allows the theory to represent various levels of these gestures. For example, a piece may build towards one key moment of tension with a set of smaller phrases, which each contain their own local tension and relaxation.

L&J distinguish between three types of relaxation (similarly for tension):

- a strong prolongation - which offers no source of relaxation for the listener, e.g. the repetition of the same event;

- a weak prolongation - which provides some sense of relaxation, e.g. the occurrence of a similar event but in a more consonant form;

- progression - this is the most relaxing kind of transition, e.g. a step to a different and more consonant event.

These six types of prolongation structure can be used to build a representation of how the events of a piece relate to each other in terms of their relaxing or tensing effect. As with the other parts of GTTM, the rules to construct a prolongational structure are presented as a mix of well-formedness rules to specify valid structures and preference rules to select from those structures those that apply to the current piece.

IRM does not treat tension/relaxation as a separate stage in the analysis, but includes it in the basic operations, for example the third condition which may lead to melodic closure is stated as: “when consonance resolves dissonance”. The extra-opus influence, i.e. influences from past experience of an idiom, must also contribute to what a listener interprets as a tensing/relaxing act. For example, if previous encounters of pieces from a specific genre included a certain pattern of events, and the current piece begins to follow that pattern but then diverges from that pattern, a feeling of tension will arise.

GCTMS does not deal with tension or relaxation in its current form.

2.4 Performance Tracking

One of the areas of research which has attracted the most attention is the problem of score matching. This can mean one of two things, either real-time tracking of the position of a performer in a musical score or trying to find a mapping from performed events to the score subsequent to the performance. The latter is generally considered to be easier than the former as the analysis system has access to the entire performance data, whereas the real-time system only has access to events up to and including the current point in the performance.

Score following is a task that can be accomplished quite easily even by an inexperienced listener. Such a listener may not be able to follow the piece continuously, but may use certain features of the score and piece (such as a long melodic rise) to correct any score following errors. At the time of writing, the available systems do not tend to use such high-level tactics to aid their tracking.

As with many problems, what seems to be a straightforward task for a human to solve is actually quite difficult for a computer. There is the initial problem of obtaining the observation of the performed event. This will inevitably involve complex mathematical operations (to detect pitch and other relevant information) which are only now beginning to be accomplished with sufficient speed to be useful. A common way around this problem is to require the musicians to use instruments that produce MIDI output. MIDI (Rothstein, 1992) is a travelled protocol designed to control electronic instruments, and has since been adopted as one of the standard ways of storing information about pieces despite its various restrictions. If the instruments used produce a MIDI output, the timing and pitch information can be taken directly from their MIDI output.

Once the event has been identified, assuming it has been identified correctly, the next problem is matching it to a place on the score. It is here that most of the difficulties lie.

A musical score is almost never a strict specification of the piece. It is usually left up to the performer to alter, or indeed improvise, the performance in ways that she feels appropriate. There are four main problems which face any score matching system (some of which are illustrated in Figure 2.1):

Missing information - The score may not precisely describe all the events which are to be performed. For example, it may leave certain areas free for improvisation or may indicate that an *ornament* should be used;

Expressive performance - An expressive performance of the piece will lead to the manipulation of, amongst other things, timing and loudness of the performed notes;

Deviations from the score - The musician may not perform the piece exactly as notated in the score. For example, in Baroque music the ornamentation is not explicitly notated in the score but it is common practice to perform the piece with rich ornamentation;

Simultaneous events - Although some events in the score may be notated as occurring at the same time, when the piece is performed they may occur at slightly

different times. For example, an *arpeggio* where notes of a chord are performed sequentially instead of simultaneously.

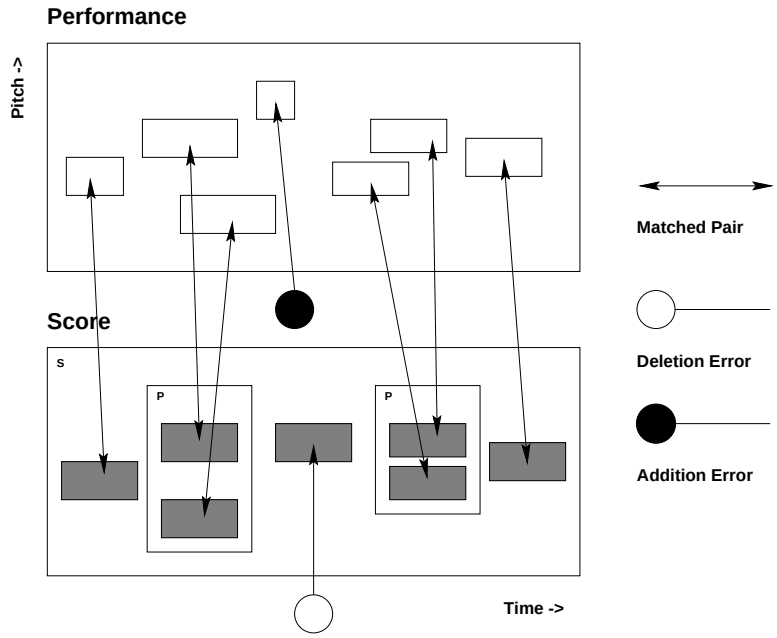


Figure 2.1: Piano-roll notation of some typical matching problems. The S indicates sequential events and the P indicates parallel ones. Adapted from Desain et al. (1997).

Desain et al. (1997) describe three different types of system which are used to perform score matching. The first is the strict matcher. This system uses a sliding window technique to sequentially match performance data with the score. A window is moved along the score and the system attempts to match every performed event with an event in the score lying within the window. This system works well with expert performances where addition or deletion errors occur only occasionally.[12]

The second type is that of a tabular matcher. These types of system take all of the performance and score data into account when trying to perform the matching. A typical system will generate an $n \times m$ table for the n performed events and the m score events. Each possible combination of score event and performed event is algorithmically assigned a fitness. A path is then found through the table which maximises these

[12] addition: the performer plays a note that does not occur in the score; deletion: the performer does not perform a note that is in the score.

fitness values. This technique is comparable to dynamic programming. See Dannenberg and Mukaino (1988) for a system that is a hybrid of the two approaches mentioned above.

Finally, there are structural matchers. These systems exploit certain structural characteristics of music to improve their capabilities. For example, during an expressive performance the timing and ordering of notes in a chord may change, however the ordering of the notes in a melody will not. The system tries to predict, using timing information, when each note in the score will occur in the performance. A window is then placed at that point in the performance and any matching events are identified. If there is more than one matching event, the system keeps multiple sets of the possible matches.

Another type of matcher is described in Grubb and Dannenberg (1997). It uses a probability distribution to track the position of a vocal performer through a score. Figure 2.2 shows the kind of distribution which can be expected during a performance. The area of the curve over the score is always 1 and the peak occurs at the most probable position of the performer. The system uses three pieces of information to track the performer:

The source position - the performer's location at the time of the previous observation;

The estimated distance - the estimated distance along the score since the last observation (calculated using estimated tempo and time elapsed);

The observation - the current measured observation.

These are used to give an estimate of the performer's current position. The system they describe does not accurately model the probability function and makes simplifications whenever possible to decrease the computational complexity of the system.

2.5 Duet Performance

Performance in a duet context is distinct from solo performance because the two musicians need to cooperate in order to generate the performance. Appleton et al. (1997)

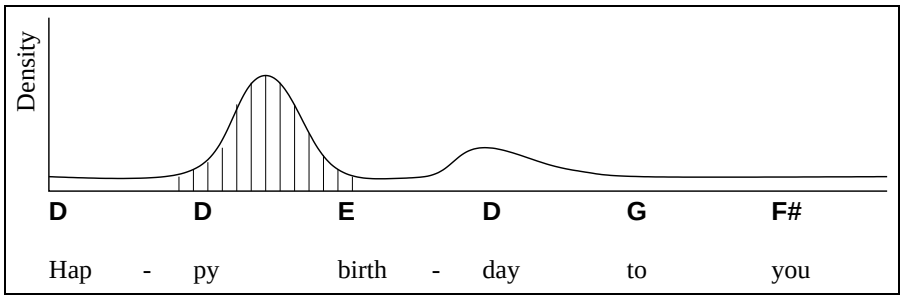


Figure 2.2: The shaded region gives the probability that the performer is singing the second note. Adapted from Grubb and Dannenberg (1997).

present the results from a very interesting experiment that looked at two different aspects of duet performance, namely the rôle of visual feedback and the effect of playing against pre-recorded performances.

It was shown that visual feedback provided an important part of the duet performance which affected the “accuracy and expressive freedom” (Appleton et al., 1997 p.474) of the performances. Where there was no visual contact between the performers, the synchronisation of their performances took place at a higher metrical level.

The results of playing against a pre-recorded performance were also documented. The performers were able to synchronise their performances more when they played against each other rather than playing against the recorded performance, despite the fact that the recorded performance will have unchanging timing. The conclusion from this part of the experiment was that the ability of the performers to cooperate with one another is an important aspect of duet performance because it allows the performers to interact and adjust to one another. When performing against the pre-recorded performance, the live musician is forced into taking a more passive rôle and made to follow the recording.

2.6 Summary

This chapter presented an overview of work related to the research presented in this thesis. Four key areas were mentioned that have a direct bearing on the research in this thesis.

Expressive Performances This section presented research about expressive performance and its relation to musical structure. It began by discussing the properties of an expressive performance, focusing specifically on the timing aspects. Importantly, it was established that although an expressive performance contains deviations from the timing notated in the score, the timing deviations for a particular pairing of piece and musician remain similar for all performances of that piece. A number of theories that attempted to generate expressive performance using a wide range of techniques were discussed. The majority of these techniques made use of the musical structure of a piece to generate an expressive performance.

Musical Structure In this section, three theories of musical structure were presented and comparisons, where applicable, were drawn between them. Each theory attempts to model similar concepts, but from a slightly different perspective. Lerdahl and Jackendoff base their theory of musical structure (GTTM) on what an experienced musical listener would perceive. Narmour takes a mixed approach with IRM and bases the bottom-up analysis on general musical principles which he claims are innate to humans and then makes use of top-down analyses which are based on past, idiom specific, experience. Finally, Cambouropoulos bases his theory (GCTMS) on general cognitive principles which are not specific to any particular genre.

GTTM and GCTMS are both presented in a manner which is conducive to creating a computer-based model around them; in fact, most of GCTMS has been implemented as a computational model by its author. IRM, on the other hand, is less amenable to a computer-based implementation.

The one principle weakness of GTTM and IRM, which is addressed comprehensively in GCTMS, is the lack of a specific definition of parallelism which plays a significant rôle in the creation of musical structure. Beyond this, both GTTM and IRM

have experimental data to support them, whereas GCTMS, because it is relatively new, has not.

Performance Tracking A number of different approaches to performance tracking were presented. All of which had to manage with typical performance errors such as omitted musical events. The approaches varied from real-time approaches to off-line ones which had the benefit of the complete performance to work with.

Duet Performance In this final section, the need for cooperation between performers in a duet context was established. The research highlighted how when a live performer plays against a predefined performance with static timing, even if that timing is expressive, the resulting performance is less satisfactory than when the two performers are able to interact.

The results for these four key areas of research suggest that in order to generate an expressive performance (as set out in Chapter 1), the system will:

1. need to have a theory of musical structure;
2. need to base its musical structure on that of the other performer;
3. use the musical structure to generate an expressive performance.

A computer-based accompaniment system, which has these three properties, is outlined in the following chapter. The components of this system provide a framework for the presentation of the research in this thesis.

Chapter 3

System Overview

This chapter presents an overview of the architecture of a cooperative performance system which provides a framework within which the research in this thesis is presented. The system consists of four distinct components, two of which are investigated in this thesis. The rôles of each of the four components, how they interact and some relevant areas of research are discussed.

3.1 Introduction

This chapter establishes a framework for a cooperative performance system. The system is designed to be modular, with each component performing a specialised task (such as structural analysis or performance tracking). This makes the system flexible enough to allow each of these modules to be replaced at a future date by a module which performs a similar task but is based upon a different theory.

The system tailors itself for each combination of musical piece and musician it encounters. Making use of rehearsal time with a particular musician enables the system to gather information on how that musician performs the current piece. This information can then be used to produce a structural analysis of the piece that corresponds to the structure implied by the performance. The structural analysis is then used as part of a process to generate a structurally based performance to accompany the current musician.

For clarity, some terms that are used in this chapter are defined below:

Current Piece - The current piece, is the duet that is the current focus of attention for the system.

Current Musician - The current musician is the human performer alongside whom the system is performing.

Current Performance - The current performance is the performance to which the system is generating a real-time expressive accompaniment.

Rehearsal Database - A database which holds recordings of the previous times that the current musician has performed the current piece.

The system is aimed towards analysing and producing expressive timing deviations. A similar set of components could be developed to deal with changes in dynamics, however timbral analysis is beyond the scope of this design.

3.2 System Components

There are four main components of the system as shown in Figure 3.1:

Performance Analysis component takes a set of rehearsal performances and identifies important features in them.

Structural Analysis component takes the musical score, and input from the performance analysis to create a structural model of the current piece for the current musician.

Prototype Performance Generation component generates a prototypical performance from the results of the structural analysis, which is partially parameterised with information from the performance analysis.

Real-time Adaptation component finalises the parameters of the prototypical performance to match the current performance and then generates the corresponding performance of the accompaniment.

Each of the four components relates to distinct areas of research. One of the goals of this work is to combine theories from the relevant fields to produce an accompanist system that has an understanding of musical structure.

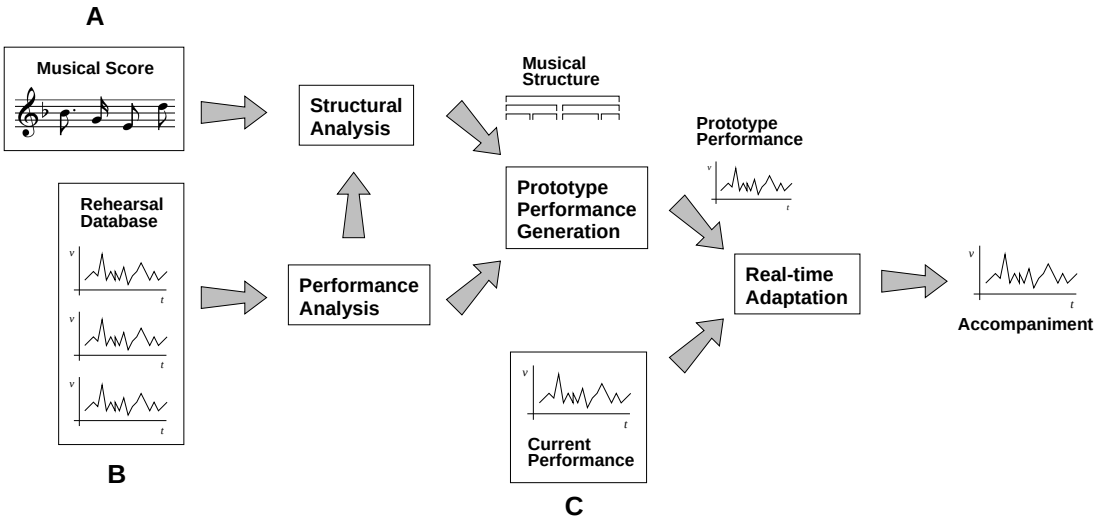


Figure 3.1: Diagram showing an overview of the system.

For the purposes of this research, the first two components are investigated. There are a number of possible theories, discussed below, which could be implemented to provide the basis to the remaining two components.

The following section describes how the system components interact to produce the accompaniment.

3.2.1 Component Interaction

There are three main inputs to the system (as shown in Figure 3.1): the musical score (A), the rehearsal database (B) and the current performance (C). The first two of these inputs are used by the system, whilst not accompanying the current performer, to analyse the musical structure of the current piece and to generate a parameterisable prototype performance. The last of these inputs, the current performance, is used to finalise the parameters of the prototype performance and enable the system to generate a suitable accompaniment for the current performer.

The database of previously recorded performances is analysed by the performance analysis component in order to identify features of the recorded performances which give clues to the musical structure of the piece. This set of features can then be passed to the structural analysis module to enable it to select the musical structure which is most strongly supported by both its implemented theory of musical structure and the set of features supplied by the performance analysis component.

The performance analysis component also produces as output a set of parameters describing how the current musician emphasises the musical structure. For example, one musician may emphasise the ending of a four-bar phrase by slowing down sharply at the end of the fourth bar; another might slow down more gradually over the last two bars of the phrase. This sort of information contributes to the shape of the prototypical performance.

The other source of information for the prototypical performance comes from the structural analysis of the piece. If the current piece has been performed in such a way as to convey the musical structure, then the musical structure has a rôle in deciding how the piece is performed. The combination of the musical structure and the information about how the performer conveys aspects of the structure can then be incorporated to produce a prototype performance to accompany the current musician.

The performance is only considered a prototype at this stage because it needs to be adjusted to match how the current musician is currently playing the piece. For example, if the musician was playing the piece at a slightly higher tempo than was previously done, then the system needs to adjust the prototype performance in order to compensate.

The following sections describe the rôle of each of the main components of the system in more detail.

3.2.2 Performance Analysis

Input:	Rehearsal Database.
Process:	Apply analyses to identify aspects of the performances that may suggest structural features.
Outputs:	Set of structural features obtained from analysis. Parameters for prototype performance generation.

The performance analysis component examines the previous performances of the current piece in order to identify features of the performances that may give clues as to the musician's interpretation of the structure of the piece.

This component has access to all previous performances of the current piece by the current musician and uses this database of performances to discover important traits in the performances that may give clues as to the current piece's structure.

There are a number of advantages to basing the analysis on a database of previous performances rather than solely on the current performance. Firstly, storing the database of performances allows for computationally intensive processes, which could not run in real-time, to be applied to the database. Secondly, by acting on stored performances, the system has access to a set of complete performances of each piece rather than just the sections of the piece that have been played so far by the current performer.

Finally, by having a set of stored performances, it is possible to minimise the amount of effect an individual performance may have on the overall results. For example, if in one particular performance of the current piece, the musician made a performance error that altered the timing pattern of a relatively unimportant musical event, the system may treat that event as being structurally significant. By having a set of performances, the effect of that error on the resulting structural interpretations will be diminished.

The main weakness of this approach is that the system will be relatively inflexible if the musician changes their understanding of the musical structure and, as a consequence, changes the way they perform the piece. If this occurs, the system will only slowly alter its structural analysis as the database of performances begins to store these new performances. A solution to this would be to adjust the relative importance of the stored performances so that the most recent performance has greater influence than

older performances.[1]

To counter this problem, a performance analysis module could be developed that works in real-time to resolve structural features (e.g. Stammen and Pennycook (1994)). This would allow for the dynamic creation of a model of the musical structure which would enable the system to adapt to whomever is currently playing the piece without the need for rehearsal time. However, the advantages, stated above, of using a database of previous performances would be lost.

The specific details of the database oriented performance analysis technique used in this research are given in Chapter 7. Briefly, the performance analysis module examines the performances for instances of phrase-final lengthening and uses these occurrences to identify the boundaries between phrases.

Once the system has analysed the rehearsal performances, it can provide data to the structural analysis component (see Section 3.2.3). The structural analysis component must be capable of producing an structure that corresponds to the way the piece was performed. The data from this performance analysis component helps choose an appropriate analysis from the many which are typically generated by theories of musical structure. For example, a piece of music might support either a four-bar or two-bar phrase structure; analysing the way the piece is performed can identify which of these phrase structures the musician was trying to convey to the listener.

3.2.3 Structural Analysis

Input:	The musical score and a set of performance derived structural features.
Process:	Apply theory of musical structure using performance derived structural features to aid the analysis.
Output:	A structural description of the piece.

The structural analysis component applies a theory of musical structure to a musical score in order to generate a set of possible musical interpretations for that piece. The component then uses the input generated by the performance analysis module to dis-

[1] Although this may give too much emphasis to a recent mistake.

ambiguate this set of possible structures to produce the one that most closely matches the way the piece is performed.

Because a piece of music can be interpreted in many different ways by a musician, the structural analysis module initially creates a set of possible musical structures for that piece rather than a single analysis.[2]

There were two main candidate theories of musical structure which were considered for this module: Lerdahl and Jackendoff's GTTM (Lerdahl and Jackendoff, 1983) and Cambouropoulos's GCTMS (Cambouropoulos, 1998).[3]

Although Cambouropoulos's GCTMS (see Section 2.3, p. 16) is presented as a style independent and a cognitively plausible theory of musical structure, it is relatively new and has not been subjected to the same amount of experimental verification as GTTM. Because of this and because GTTM is a more complete theory (see Section 2.3, p. 16), it was chosen as the theory of musical structure to be implemented. However, as expressed in the introduction to this chapter, the system has been designed in such a way as to allow an implementation of GCTMS (or another similar theory) to be used in its place.

Once the structural theory is applied to the musical score, the resulting set of possible structures needs to be narrowed to the structure that most closely resembles the way the piece is being performed. To do this, the results from the performance analysis component are applied to the set of possible structures to select those that have the structural features identified from the database of performances. For details of this process see Chapter 9.

3.2.4 Prototype Performance Generation

Inputs:	Structural analysis of the piece and its musical score.
	How the structure is conveyed by the performer.
Process:	Apply structure based model of performance.
Output:	A parameterisable prototypical performance.

[2] Provided that the theory of musical structure supports multiple structures.

[3] Only theories of musical structure which can be expressed in a rule-based form are suitable for this component.

The performance generation component takes as input a structural description of the piece, its musical score and information about the musician's performance technique (i.e. how the structure is conveyed through the performance). The performance generation component then applies a model of musical performance, parameterised with the musician's performance technique, to generate an expressive performance.

As with the structural analysis component, there are a number of different theories of expressive musical performance which could be incorporated as part of this module.

Gerhard Widmer's (2000) research into expressive musical performance results in a set of machine-learnt rules that take a musical score and a structural representation of that score and adjust the performance parameters accordingly. Widmer's system takes as input a structural representation of a piece, the score of that piece and a performance of that piece. From these three inputs, the system derives a set of performance rules based upon the relationship between the structure, score and performance. Using this technique, the performance generation component described here would not need to take the musician's performance technique as input because this would be derived automatically.

Friberg et al. (1997) uses a rule-based system which bases its performance on both structural features and the musical score. It is similar in nature to Widmer's research but it is based on human derived rules and parameters.

Todd's model, based on the idea of phrase-final lengthening takes a score and structural analysis of that score and applies a parameterisable hierarchy of parabolic curves to the mechanical timing of the piece to generate an expressive performance. In Todd (1985), the Time-Span Reduction component of GTTM is used as the basis for the structural representation of the piece.

The three models presented above are not a complete list of models of musical performance, they are, however, the theories that would be most suited to being incorporated into the current architecture.

3.2.5 Real-time Adaptation

Input:	A parameterisable performance of the piece.
	The current performance of the piece.
Process:	Analyse current performance and apply appropriate parameters.
Output:	Adapted accompaniment.

The parameterisable performance generated by the previous component provides a basis for an expressive performance, however each performance of a piece is individual so it is important for the system to be able to adapt to the current performance of the piece.

This module needs to 'listen' to the current performance and identify important local and global features such as the tempo, loudness, etc. If the prototype performance is viewed as a parameterisable function, this module needs to derive and supply the parameters for that function.

Another important behaviour of this component is to provide synchronicity between the human musician and the system. At significant points of the performance the component needs to identify the human performer's position and adjust as appropriate. This sort of synchronicity is normally achieved through audio or visual clues between performers (Appleton et al., 1997).

There are two aspects of this component: the tracking of the human performer's position in the score, and the derivation of the parameters to be applied to the prototypical performance. The derivation of the performance parameters relies upon the system being capable of comparing the expected performance with the actual performance by the human musician and adapting accordingly.

In order to allow for this comparison between the expected and actual performances, the system needs to be able to follow the human musician and to match individual events between the expected and actual performances. Roger Dannenberg has made significant contributions to this field (e.g. Dannenberg and Mukaino (1988); Dannenberg (1993)) and recently developed a system along with Lorin Grubb (1997) that tracks and accompanies a vocal performer using statistical methods.

Another body of work by Desain et al. (1997) has focussed on using properties of

music, such as the relatively strict order of events, to assist with the matching process.

If either of these were incorporated into the system, they would allow the current performance to be monitored and the system could then generate a set of parameters for the parameterisable performance.

3.3 Summary

This chapter presented an overview of the system that would result from an implementation of the main ideas presented in this research. The system consists of four main components, the first two of which are the focus of this work:

1. The performance analysis component will take as input a database of previous performances of the current piece and identify features in the performance which may give clues towards the structural properties of the current piece. This component is presented in Chapter 7.

2. The structural analysis component takes as input the musical score and the results from the performance analysis component. A theory of musical structure is then applied to the score to obtain a set of candidate musical structures. The structure that most closely resembles the way the piece was performed is chosen by incorporating the information from the performance analysis. This structure, it is argued, corresponds to the structural interpretation of the current piece by the performer. The structural analysis component is presented in Chapter 4 and the interaction between this component and the performance analysis component is presented in Chapter 9.

A working implementation of the performance generation and the real-time adaptation components are not presented in this thesis. However, a brief overview of research relevant to a full implementation of the described system was presented along with an outline of how they would interact with the rest of the system.

Chapter 4

Structural Analysis

This chapter[1] outlines the functionality and properties of the structural analysis module (see Chapter 3). The module takes a suitably encoded musical score as input and generates a set of constrained possible grouping boundary points.

4.1 Introduction

This chapter presents a new implementation of Lerdahl and Jackendoff's (1983) Grouping Rules which are the basis for the Structural Analysis component. Figure 4.1 shows how this component interacts with other components in the structural disambiguation flow (which corresponds to the interaction of the two components labelled Structural Analysis and Performance Analysis in Figure 3.1).

The structural analysis component takes as input a representation of a musical score using Common Hierarchical Abstract Representation for Music (*Charm*) (Smaill et al., 1993) and finds all possible positions at which the grouping rules can apply. These positions correspond to potential boundary points in the grouping analysis.

A boundary can only occur between two events and is used to indicate that the two events either side of the boundary belong to separate groups. These boundaries act as a specification for the set of possible musical structures which, according to GTTM, could be applied to the piece of music.

[1] Some of the material in this chapter has already appeared as Curry and Wiggins (1999)

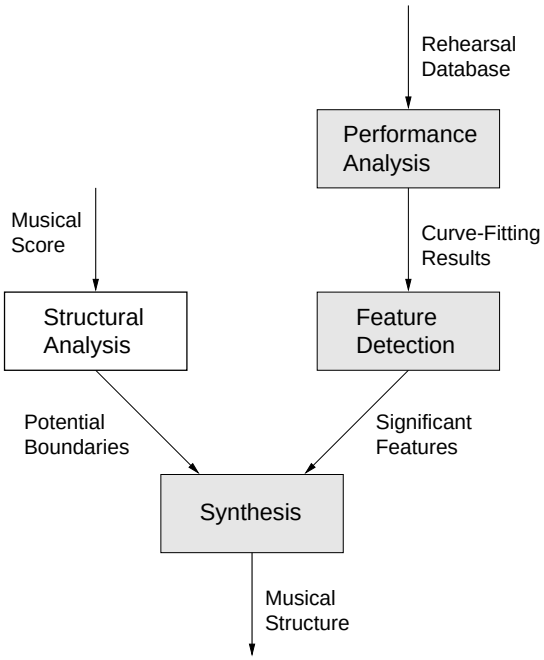


Figure 4.1: Pictorial representation of the rôle of this component within the structural disambiguation flow.

The following sections recapitulate the structure of GTTM and then describe the Grouping Structure component in more detail.

4.2 The Generative Theory of Tonal Music

The authors of GTTM state its goal to be a "*formal description of the musical intuitions of a listener who is experienced in a musical idiom*" (Lerdahl and Jackendoff, 1983 p.1). GTTM attempts to formalise the process of how an 'experienced listener' perceives the structure of the piece being heard. The theory itself is divided into four sections which can be broadly grouped into two different categories.

The first describes rhythmic structure which encompasses both the grouping structure and the metrical structure of a piece. The second group is based upon the notion of reduction in which some musical events are structurally less important than others. Specifically the four structural concepts are:

Grouping Structure is the segmentation of musical events into groups of similar or related events. These rules try to encapsulate the notion of chunking in which a listener groups certain events together whilst hearing the piece.

Metrical Structure models the relative strength and weakness of events at various levels in a metrical hierarchy. It captures the notion of strong and weak beats.

Time-span Reduction identifies pitch events which are perceived as being of greater structural importance at various levels.

Prolongational Reduction identifies events which represent the harmonic movement of the piece. The prolongational reduction deals with issues such as tension, relaxation, continuity and progression.

For the purposes of this research, this module has been based solely upon the grouping structure aspects of the theory. The grouping structure is the first step in providing a complete analysis and is the foundation on which the other analyses of the theory build. A full implementation of GTTM is a desirable future goal and could be incorporated into this work to both aid the resolution of the grouping boundaries (because

the grouping structure interacts with the time-span and prolongational reductions) and to provide a more complete interpretation of the musical piece.

The following sections introduce the rules which define the grouping structure of a piece and describe their implementation in an automated system.

4.3 Grouping Structure

The grouping rules provide a formal description of how grouping structures can be created from analysis of the musical surface. The rules can be sub-typed into three different categories:

Well-formedness Rules which state which structural descriptions are possible.

Preference Rules which try to select from the possible structures the ones that correspond to what an experienced listener would hear.

Transformational Rules that allow certain distortions of the strict structures prescribed by the well-formedness rules.

The rules corresponding to each category as regards grouping structure are presented below including a brief example of their application.

4.3.1 Well-formedness Rules

The first set of rules presented by Lerdahl and Jackendoff[2] prescribe what structures correspond to a valid grouping structure.

GWFR1 Any contiguous sequence of pitch-events, drum beats, or the like can constitute a group, and only contiguous sequences can constitute a group.[3]

This first rule specifies that a group has to consist of a set of neighbouring musical events. It prevents a group from consisting of, for example, every other musical note of a piece.

[2] From this point onwards 'Lerdahl and Jackendoff' will usually be abbreviated by L&J.

[3] All the definitions of the grouping rules are taken directly from Lerdahl and Jackendoff (1983 pp. 36-67).

GWFR2 simply specifies that an entire piece is a group and should be viewed as a related collection of musical events. GWFR3 allows a group to contain other groups and therefore introduces the notion of a hierarchical structure of groups.

GWFR2 A piece constitutes a group.

GWFR3 A group may contain smaller groups.

Finally, grouping well-formedness rules four and five impose some stricter conditions on the nature of the possible group hierarchies. The first of these, GWFR4, prevents a sub-group from being shared between two higher level groups. GWFR5, forces all of the grouping structure to be segmented to the same depth with every event belonging to a group.

GWFR4 If a group G_1 contains part of a group G_2, it must contain all of G_2.

GWFR5 If a group G_1 contains a smaller group G_2, then G_1 must be exhaustively partitioned into smaller groups.

Figure 4.2 shows an example of two grouping structures that are not considered well-formed under the above rules. Example (a) in the figure shows a structure that breaks rule GWFR4: The group *G*2 is not wholly contained within group *G*3 - they partially overlap. Example (b) shows a structure that breaks rule GWFR5: Although both groups *G*2 and *G*3 are contained within *G*1, those two groups do not completely partition the space claimed by *G*1 - there is a gap between the two groups at the lower-level.

Figure 4.2: Illegal grouping structures that contravene (a) rule GWFR4 and (b) rule GWFR5.

The well-formedness rules specify a hierarchical structure that can be used to represent a grouping of objects together. The preference rules take this set of well-formedness rules and provide a means of applying them to a *musical surface* in such a way as to represent the musical structure of that specific surface.

4.3.2 Preference Rules

The preference rules are applied to select, from the possible structures generated by the well-formedness rules, those structures that correspond to what an experienced listener would hear. The preference rules are based on Gestalt principles of similarity and difference. Lerdahl and Jackendoff draw parallels between the auditory and visual grouping of events under the basic principles that similar events are grouped together and events that differ are separated.

The following sections present the two different types of preference rules: the local detail rules which focus on identifying low-level structural boundaries and the larger-level grouping rules which attempt to model higher-level grouping boundaries such as phrase beginnings and endings.

4.3.2.1 Local Detail Rules

The first grouping preference rule (GPR1) expresses the desire for as few small groups as possible. The goal of the analysis process is not to segment the piece exhaustively, rather to identify groups of events that have a musically salient reason to be gathered together. The rule does not preclude the creation of small groups which may be desired in some situations, but a preference is established for larger groups.

GPR1 Strongly avoid groups containing a single event.

GPR1, alternative form Avoid analyses with very small groups – the smaller, the less preferable.

The second preference rule (GPR2) is the first of the rules that makes use of information from the musical surface. The rule is based on the concept of proximity, in this case proximity in time rather than space. Events that are more separated from one

another in musical time than they are from their neighbouring events are assigned to different groups.

GPR2 (Proximity) Consider a sequence of four notes $n_1n_2n_3n_4$. All else being equal, the transition $n_2 - n_3$ may be heard as a group boundary if

a. (Slur/Rest) the interval of time from the end of n_2 to the beginning of n_3 is greater than that from the end of n_1 to the beginning of n_2 and that from the end of n_3 to the beginning of n_4, or if

b. (Attack-Point) the interval of time between the attack points of n_2 and n_3 is greater than that between the attack points of n_1 and n_2 and that between the attack points of n_3 and n_4.

GPR3 uses the concept of similarity to identify groups of events that belong together. Specifically it expresses the notion that when presented with a stream of events, any pair of events that is less similar to one another than they are from their respective neighbouring events causes a potential grouping boundary. Using L&J's notation, given four notes $n_1n_2n_3n_4$; if n_2 and n_3 are less similar than n_1 and n_2 and less similar than n_3 and n_4 then the transition from n_2 to n_3 may be considered a boundary point. GPR3 lists four specific cases which may cause this sort of boundary but L&J comment that there are probably other qualities of musical events, such as timbre, which could cause such boundaries.

GPR3 (Change) Consider a sequence of four notes $n_1n_2n_3n_4$. All else being equal, the transition $n_2 - n_3$ may be heard as a group boundary if

a. (Register) the transition $n_2 - n_3$ involves a greater *intervallic* distance than both $n_1 - n_2$ and $n_3 - n_4$, or if

b. (Dynamics) the transition $n_2 - n_3$ involves a change in dynamics and $n_1 - n_2$ and $n_3 - n_4$ do not, or if

c. (Articulation) the transition $n_2 - n_3$ involves a change in articulation and $n_1 - n_2$ and $n_3 - n_4$ do not, or if

d. (Length) n_2 and n_3 are of different lengths and both pairs n_1, n_2 and n_3, n_4 do not differ in length.

The above three rules are used to detect local level grouping boundaries. They contain no reference to the construction of the higher-level grouping structures allowed by the well-formedness rules. The next section presents the rules used to create these higher-level structures.

4.3.2.2 Organisation of Larger-Level Grouping

The intensification rule (GPR4) allows the local level preference rules to contribute to the higher-level structure by noting that the resulting boundary strength of the rules can vary depending upon the context of their application. For example, in Figure 4.3 the rest between the third and fourth triple of events causes a relatively more pronounced grouping boundary to be applied, thus introducing another level into the grouping hierarchy that clusters the first three and the last two groups together.

GPR4 (Intensification) Where the effects picked out by GPRs 2 and 3 are relatively more pronounced, a larger-level group boundary may be placed.

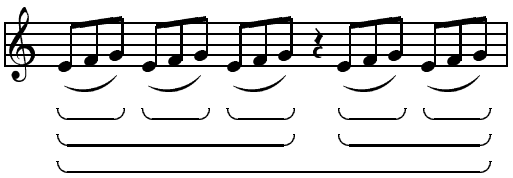


Figure 4.3: An example of GPR4 (Intensification), the higher level grouping boundary is created due to the extra contribution of the rest. Adapted from Lerdahl and Jackendoff (1983 p. 49).

The fifth preference rule (GPR5) expresses the idea that a symmetrical grouping is preferred over a non-symmetrical one. The rule is used to give the higher-level groupings a balanced structure.

GPR5 (Symmetry) Prefer grouping analyses that most closely approach the ideal subdivision of groups into parts of equal length.

Figure 4.4 shows examples of the results of applying the symmetry rule. The first example (a) shows how GPR5 has created a stable binary hierarchy over the four small groups of events rather than, for example, clustering the first three groups together and leaving the last as a singleton.

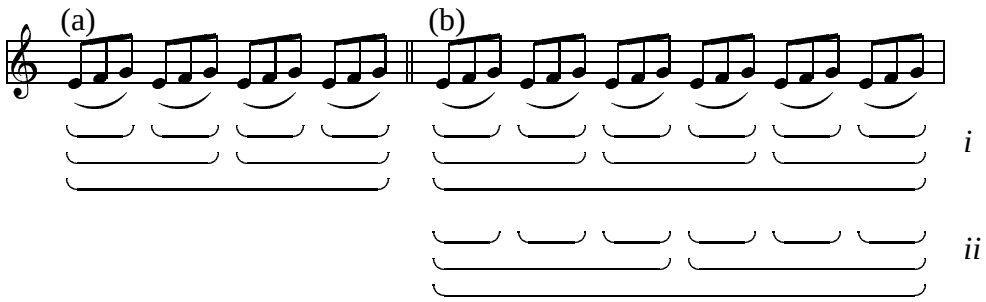


Figure 4.4: An example of the effects of the application of GPR5 (Symmetry). Excerpt (a) shows a stable binary structure. Excerpt (b)'s groupings *i* and *ii* show the conflicts arising from a ternary structure. Adapted from Lerdahl and Jackendoff (1983 p. 50).

The second example (b) shows how the ternary structure of this excerpt causes instability in the grouping results. Both grouping hierarchies *i* and *ii* are valid structures and indeed, both conform to GPR5. In hierarchy *i* the effect of the symmetry rule has been applied to the lowest-level[4] of the grouping structure which has resulted in three intermediate groups. Grouping hierarchy *ii* shows the results of applying GPR5 to the intermediate level which results in two larger level groups of three lower-level groups. In a real piece, this ambiguity may be resolved using other information such as parallelism (see below).

GPR6 introduces the notion of parallelism and states that if a piece has a repetitive structure, then those parts of the piece that form that repetitive structure should, preferably, be assigned similar grouping structures. This is one of the theory's most powerful, but least defined, grouping rules.

GPR6 (Parallelism) Where two or more segments of the music can be construed as parallel, they preferably form parallel parts of groups.

[4]Closest to the musical surface.

The ability to identify similar passages of music is notoriously difficult for computerised systems to perform. In order to find these similar passages, a means of measuring how similar two sections are has to be defined.

For example, Figure 4.3 shows an example consisting of five sets of three events. Each of these sets is identical in timing and pitch and so they are easily be identified as being parallel to each other. If the events were not identical, there would have to be some measure of similarity that could return a numerical value corresponding to how different two segments were and also a threshold value to decide whether the two segments could be considered parallel or not. Cambouropoulos et al. (1999) provides a good overview of the issues involved in comparing two sets of musical events and summarises a number of different approaches to this problem.

The final grouping preference rule (GPR7) ties together this part of GTTM with the Time-Span and the Prolongational Stability reductions described in Chapter 2.

GPR7 (Time-Span and Prolongational Stability) Prefer a grouping structure that results in more stable time-span and/or prolongational reductions.

Together, these well-formedness and preference rules offer a means of producing grouping structures for a large portion of the possible musical streams. However, there are some exceptions that appear in music which are not supported by the well-formedness rules presented above. These exceptions are handled by the transformational rules.

4.3.3 Transformational Rules

The transformational rules allow for two special situations which allow grouping structures resulting from the application of the well-formedness and preference rules to be transformed to more closely represent the intended musical structure.

Grouping Overlap - a grouping overlap occurs whenever one event, or set of events, forms the end of one group and the start of another. In this case, the event can be treated as two distinct events with the same properties but with one corresponding to the end of the first group and the other as the start of the second.

Grouping Elision - an elision is similar to an overlap, it occurs whenever two groups can be perceived as sharing an event, or set of events, however the event belongs more strongly to one group or the other.

The rules are added as new transformational rules rather than adaptations of the well-formedness rules to prevent the dilution of the well-formedness rules (specifically GWFR4 and GWFR5) due to the relative rarity of the situations which require the application of the transformational rules.

4.4 Musical Representation

The musical pieces are represented using Common Hierarchical Abstract Representation for Music (*Charm*) (Smaill et al., 1993; Wiggins et al., 1993). *Charm* is proposed as a first step towards a musical representation system that is independent of style, tonal system etc.

There are two components of the *Charm* abstract data type, they are: *events* and *constituents*. *Events* provide the representation of basic musical events. They represent information such as pitch, onset time, duration and timbre. *Constituents* provide a means to group structurally related events together.

For the purposes of this research the musical events are represent by a Prolog fact of the form:

```
event(<id>, <pitch>, <onset-time>, <duration>, <timbre>)
```

where:

`<id>` is a unique identifier for this event;

`<pitch>` is a representation of the musical pitch, in this case a triple of note, accidental and octave;

`<onset-time>` the starting time for the musical event, given in terms of a common starting time (such as zero for the start of the score);

`<duration>` the duration of the event as notated in the score;

`<timbre>` allows for the addition of timbral information for this event.[5]

The *constituents* allow for additional information to be recorded that relates to groups of events rather than individual ones. The prolog fact used to represent the constituents is of the form:

```
constituent(<id>, <structural-type>, <musical-type>, <collection>)
```

where:

`<id>` is a unique identifier for this constituent;

`<structural-type>` specifies the nature of the constituent; in this case, structural-type could be one of: stream, articulation or dynamics;

`<musical-type>` a label which represents the rôle of the constituent;

`<collection>` the set of events to which this constituent applies.

Figure 4.5 shows the score of bars 3–6 of *Berceuse* by Gabriel Fauré. Table 4.1 shows the top line of this score represented using the *Charm* notation. As can be seen from the table, each of the eleven notes has been allocated a unique event instance. The # symbol in the pitch tuple represents a sharp accidental; the = symbol represents a natural accidental.

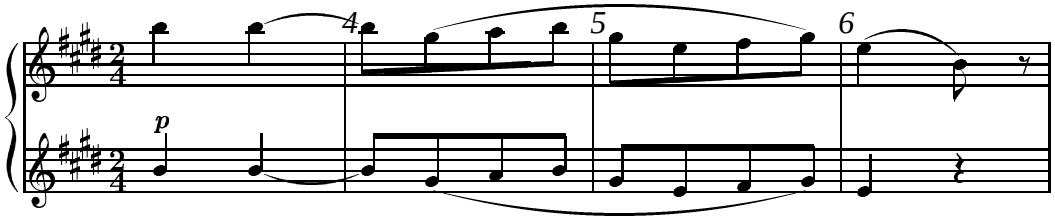


Figure 4.5: Bars 3–6 of Fauré's *Berceuse*.

The eleven events are then collected together as one continuous set of events by the use of a stream constituent (c001). The other properties of the events, such as the dynamics and articulation are similarly represented by constituents of the appropriate structural and musical types (c002, c003 and c004).

[5] In this research, the timbre field remained unused.

```
event(e001, [b, =, 6], 00, 2, []),
event(e002, [b, =, 6], 02, 3, []),
event(e003, [g, #, 6], 05, 1, []),
event(e004, [a, =, 6], 06, 1, []),
event(e005, [b, =, 6], 07, 1, []),
event(e006, [g, #, 6], 08, 1, []),
event(e007, [e, =, 6], 09, 1, []),
event(e008, [f, #, 6], 10, 1, []),
event(e009, [g, #, 6], 11, 1, []),
event(e010, [e, =, 6], 12, 2, []),
event(e011, [b, =, 5], 14, 1, []),
constituent(c001, stream, faure1, [e001, e002, ..., e011]),
constituent(c002, dynamics, piano, [e001, e002, ..., e011]),
constituent(c003, articulation, slur, [e003, e004, ..., e009]),
constituent(c004, articulation, slur, [e010, e011])]).
```

Table 4.1: Representation of the top line of Fauré's *Berceuse* bars 3–6 in *Charm*.

4.5 Implementation

This section describes how the grouping rules are incorporated into the analysis module and how the introduction of a ranking preference to the rules allows the assignment of a grouping structure hierarchy to the piece being analysed.

4.5.1 Related Work

Robbie (1994) developed an interactive system for GTTM analysis. The system took as input a musical score represented using *Charm* and allowed a user to interactively apply the grouping rules to the score. The final result of the system was a user-informed analysis of the piece.

The underlying algorithm iterates across the musical surface attempting to find occurrences where a grouping rule could apply. The algorithm, which was both effective and intuitive, has been adapted for use in this implementation by extending its ability to cope with certain rules and by incorporating the use of feature-category weights.

4.5.2 Subset of rules

The implementation of the grouping rules analysis does not explicitly include all the rules. All of the well-formedness rules are implicitly included by the combination of the tree-based representation of the grouping structure (see Chapter 5) and the notion of grouping boundaries.

The system explicitly defines grouping preference rules GPR2 and GPR3 as rules that return boundary positions based upon the musical surface. The intensification rule, GPR4, is partially included by summing the weights of the rules which are applicable at a point in the score to produce a total strength for that boundary. A more complete implementation of GPR4 would also support the assigning of strength based upon the context of the rule's application (see Sections 4.5.3 and 4.5.5).

As regards grouping preference rules GPR1, GPR5 and GPR6, although they are not explicitly represented in this analysis module, it is expected that the input from the performance analysis module (see Chapters 7 and 9) will perform a similar rôle.

Finally, the preference rule GPR7 and the two transformational rules are not represented in the current implementation of the system because they rely on a more powerful and complete implementation of GTTM before they can be applied with confidence.

4.5.3 Assigning Weights

Lerdahl and Jackendoff draw attention to the fact that they believe the rules are not all equal in significance and provide a suggestion for how this could be used in a full implementation of the rules.

> In order to make the theory fully predictive, it might be desirable to assign each rule a numerical degree of strength, and to assign various situations a degree of strength as evidence for particular rules. Then in each situation the influence of a particular rule would be numerically measured as the product of the rule's intrinsic strength and the strength of evidence for the rule at that point; the most 'natural' judgement would be the analysis with the highest total numerical value from all rule applications.
>
> (Lerdahl and Jackendoff, 1983 p.47)

L&J discuss a possible relative weighting of the rules, but fall short of providing a quantitative set of weights. They produce a number of examples illustrating the rules in conflict, and from these the following qualitative weightings can be drawn.

$$\begin{aligned} \text{GPR2(a)} &> \text{GPR2(b)} \\ \text{GPR2(a)} &> \text{GPR3(a)} \\ \text{GPR3(b)} &> \text{GPR2(b)} \\ \text{GPR2} &> \text{GPR3} \end{aligned}$$

Deliège (1987) performed two experiments to both test the validity of the rules with respect to musicians and non-musicians and to see if there was a ranking that could be applied to the rules. Briefly, the first experiment confirmed that the grouping rules were mostly indicative of listener's grouping results. This experiment showed that the musicians' grouping strategies were most similar to those proposed by Lerdahl and Jackendoff and therefore supported their claims that the theory was aimed to model what an 'experienced listener' would do.

Table 4.2 shows some of the results from Deliège's experiments. The 'Ease of Decision' column is a measure of how confidently the rules were applied by a group of musicians without the need for repeated listening to the samples. The 'Index of Stability' provides a similar measure based on whether, on repetition of the stimuli, the same rules were applied by the subjects. If these two indices are treated as an indication of a rule's strength, in terms of how easily instances of the rule can be identified and how stable the application of a rule tends to be, the table shows that there is some variation in the strength of the rules according to the two measures.

Rule	Ease of Decision	Index of Stability
GPR2(a)	0.66	0.76
GPR2(b)	0.88	0.68
GPR3(a)	0.78	0.79
GPR3(b)	0.77	0.74
GPR3(c)	0.58	0.74
GPR3(d)	0.64	0.83

Table 4.2: Musician's Ease of Decision and Index of Stability measures for the grouping rules (Deliège, 1987).

Importantly, these experiments were based on presenting the examples to the subjects as audio examples rather than as a musical score. The audio examples, although created to illustrate the rules, may in themselves affect the weightings of the applied rules. For example, the choice of instrument may affect the relative strength of the attack-point rule or the articulation rule when contrasted with the other rules.

The variation in strength for the rules is slight for most cases, but for some instances such as GPR2(b) and GPR3(d) the two indices produce contrasting results. Deliège concluded that there was a tendency for different rules to have different saliences, but that further work needed to be performed in order to state the differences with confidence.

When applying the rules to the three example pieces, it became apparent that the slur/rest rule, GPR2(a), could be better treated as two separate rules: one to deal with

slurs, the other for rests. When a sequence of events is performed which contains a rest, the stream is disrupted much more significantly than it is when two events belong to different slurred groups. Deliège's experiments into the relative strengths of the rules made use of the slur form of the rule rather than the rest. In this research, the slur/rest rule will be considered as two separate rules which have different strengths.

Table 4.3 shows the feature-category strengths that were applied to GPR2 and GPR3. Due to the relative similarity of rule-strengths and the sometimes contradictory results of Deliège's experiments, it was decided to assign weights to the rules which corresponded to L&J's qualitative description of their importance. Further to this, the added condition was applied that the application of any two rules would provide a greater strength than just one rule alone.

Although a number of different weightings were briefly experimented with before these final values were chosen, the weights have not been experimentally confirmed and are used mainly to demonstrate the ability of the system to handle weighted rules. Further work should investigate these values to a greater extent and perform some more detailed experimental research to either support or reject the current values. It may be that the weights are idiom or piece specific, in which case the assignment of weights to the rules would take on a greater level of complexity than is dealt with here.

Rule	Weight
GPR2(a)(Rest)	7
GPR2(a)(Slur)	4
GPR2(b)(Attack-Point)	5
GPR3(a)(Register)	4
GPR3(b)(Dynamics)	6
GPR3(c)(Articulation)	4
GPR3(d)(Length)	4

Table 4.3: Assignment of weights to rules according to discussion in Lerdahl and Jackendoff (1983).

The current implementation does not take into consideration the context of a rule's application. For example, if the algorithm encountered a minim rest and a semi-quaver rest, both would be assigned the same score even though it is probable that the minim rest would establish a stronger grouping boundary than the semi-quaver rest.

However, in order to identify the appropriate weighting factors, a further experiment would need to be performed which compared both the relative salience of the rules and instances of situations which strengthened their application. For example, is a quaver rest more significant than an octave sized leap in register? Is it more significant than a two-tone leap in register?

4.5.4 Switches

One of the criticisms levelled at GTTM is that it over-generates, i.e. in the case of grouping boundaries, it finds more possible boundaries than are actually used in the final grouping analysis. This can equally be viewed as one of the theory's great strengths; it allows the concept of multiple potential grouping structures which become resolved as the piece is listened to. It also supports the notion that different listeners can have a different understanding of the music and therefore have a different view of the structure.

In order to support this concept, the implementation of grouping structure includes the notion of a switch. Every potential grouping boundary has a switch which can be in one of two states: either on or off. If the switch is 'on' then that forces the potential grouping boundary at that point to be active. If the switch is 'off' then the grouping boundary is inactive and has similar consequences to assigning the boundary a grouping strength of zero and turning its switch 'on'.

As an aid to the implementation, all inter-event transitions are considered to have potential boundaries. Once the grouping rules have been applied to the data, those inter-event transitions which have no identified boundaries (i.e. where no rule applications were found) are assigned a boundary strength of zero and have their switches permanently set to 'on'. This forces those events which have no rule applications between them to be assigned to the same group.

The resulting set of switches offers a means of controlling the set of grouping

structures produced and it is with these switches that later parts of the system can affect the grouping structure (see Chapter 9).

4.5.5 Weight balancing

When the algorithm is applied to the data in Table 4.1 the results presented in Table 4.4 are obtained.[6] As can be seen, there are three grouping boundaries identified, one between events e002 and e003, one between e006 and e007 and finally between e009 and e010. The rules which are applicable are returned along with their strength as a tuple of the form:

```
rule(<ruleid>, <strength>)
```

Where `<ruleid>` is the name of the rule that applies at that point and `<strength>` is the relative strength of that rule. As discussed above, the strength of the rule is independent of context and is only a measure of the strength of that rule when compared with the other rules.[7]

Boundary Position		Rule(s)
e002	e003	[rule(gpr2b,5),rule(gpr3a,4)]
e003	e004	[]
e004	e005	[]
e005	e006	[]
e006	e007	[rule(gpr3a,4)]
e007	e008	[]
e008	e009	[]
e009	e010	[rule(gpr2a,4)]

Table 4.4: Results of the grouping algorithm when applied to the data representing bars 3–6 of *Berceuse* (as shown in Table 4.1).

[6] The event boundaries e001-e002 and e010-e011 are not shown as they are boundaries that occur at the edge of the excerpt and are not contained within the required set of four contiguous events needed for the application of the grouping rules.

[7] The rule's 'intrinsic' strength according to L&J.

The strengths of the rules that apply at each boundary are summed to produce a total measure of strength for that particular boundary. These boundary strengths provide a means by which the higher-level grouping structure can be chosen. In the current example, this would suggest a boundary strength of 9 between e002 & e003 and a strength of 4 between e006 & e007 and e009 & e010.

However, these strengths do not provide a precise measure of the boundary depth that should be applied at that point, rather they provide a relative numerical measurement of strength with respect to the surrounding boundaries. In this case, all that should be concluded is that the boundary between e002 and e003 is greater than the other two boundaries.

The relationship between switches and size of boundary strengths can be formalised as follows. Let $str_{a,b}$ represent the measured strength of the boundary between event a and event b and $sw_{a,b}$ represent the switch used to indicate whether the boundary is active or not. Then it is possible to create a constrained boundary strength $\sigma_{a,b} \geq 0$ which is dependent upon both the measured strengths and the switches as below:

$$(sw_{a,b} = \text{on}) \wedge (sw_{c,d} = \text{on}) \wedge (str_{a,b} > str_{c,d}) \Rightarrow (\sigma_{a,b} > \sigma_{c,d}) \tag{4.1}$$

Given that the on/off state of the switches is represented by 1/0, the above can be analogously expressed as follows:[8]

$$(sw_{a,b} \times str_{a,b}) > (sw_{c,d} \times str_{c,d}) \Rightarrow (\sigma_{a,b} > \sigma_{c,d}) \tag{4.2}$$

This constraint is propagated across all the possible grouping boundaries (including those that have zero strength) and the values of the boundary strengths are subsequently minimised as far as is possible given that the state of the switches is unknown at this time.

Any solution to the above equation, when applied across all the grouping boundaries, will produce the smallest possible tree, i.e. the tree with the least branching, which matches the specification given by the boundary strengths and switches. The generated tree will correspond to the grouping analysis of the piece with the connec-

[8] Where all terms are integer based and must be greater than or equal to zero.

tions between the nodes indicating the hierarchy of the grouping structure. The resulting hierarchy will be the most compact hierarchy that obeys the grouping strengths and boundaries as found by the analysis process.

Collating the above steps together gives the final results shown in Table 4.5. The table shows that the grouping boundaries have the potential to be on or off, and as a result of this, not all the boundary strengths can be decided.

Boundary Position		Switch State	Boundary Strength
e002	e003	0/1	$0/(1 \leq \sigma_{e002,e003} \leq 2)$
e003	e004	1	0
e004	e005	1	0
e005	e006	1	0
e006	e007	0/1	$0/(\sigma_{e006,e007} = 1)$
e007	e008	1	0
e008	e009	1	0
e009	e010	0/1	$0/(\sigma_{e009,e010} = 1)$

Table 4.5: Final grouping results for bars 3–6 of *Berceuse*.

Taking the particular case when all the switches are 'on', applying this constraint to the running example would give the following results:

$$\begin{aligned} \sigma_{e002,e003} &> \sigma_{e006,e007} \\ \sigma_{e002,e003} &> \sigma_{e009,e010} \\ \sigma_{e002,e003} &> 0 \\ \sigma_{e006,e007} &> 0 \\ \sigma_{e009,e010} &> 0 \end{aligned}$$

If maximum value of σ is minimised, this will return a solution of:[9]

$$\sigma_{e002,e003} = 2$$

[9] In total, depending on the settings of the switches, there are eight possible combinations of values for the σ strengths: $[(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,0),(2,0,1),(2,1,0),(2,1,1)]$.

$$\sigma_{e006,e007} = 1$$
$$\sigma_{e009,e010} = 1$$

Every undecided boundary point increases the possible solution set by a factor of two, so, even if there are only grouping boundaries for every 10% of the events in *Berceuse*, that gives a potential grouping set of 2^{26} which is approximately 67 million possible structures.

4.6 Results

This section contains a detailed discussion of the results of applying the grouping analysis system to some small musical excerpts and more general discussion of the results when applied to the three pieces used in Chapters 6–9.

4.6.1 Grouping: Excerpt from *Berceuse*

Figure 4.6 shows the results of applying the grouping algorithm to four bars of *Berceuse* containing the main theme. There are four grouping boundaries identified by the system (indicated beneath the score). All of the proposed boundaries seem musically sensible. However the third boundary, between notes *9* and *10*, seems very weak and so can be ignored in this particular analysis.[10]

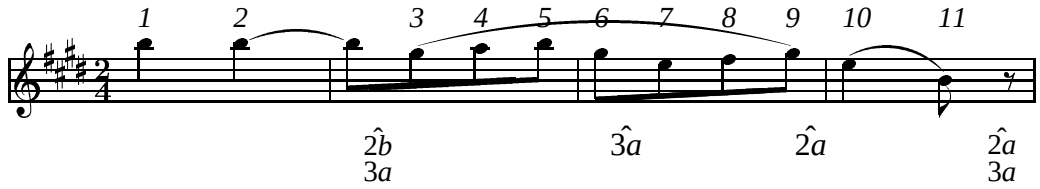


Figure 4.6: Possible grouping boundaries for bars 3–6 of Fauré's *Berceuse*.

Assuming the strengths of the rules as defined in Table 4.3 are used[11], and the grouping boundary between *9* and *10* is 'switched off', then the grouping structure in Figure 4.7 is obtained.

[10] Other people's interpretations may support the existence of this boundary. The aim of this research is to use performance analysis to make this kind of decision (see Chapter 9).

[11] Namely that: GPR2(a)+GPR3(a) > GPR2(b)+GPR3(a) > GPR3(a)

The structure highlights the fact that the four bar excerpt forms a complete phrase, which is made up of two sections. The first section contains the two long notes at the beginning of the phrase and the second section contains the remaining nine notes which are themselves subdivided into two parallel sets of notes.

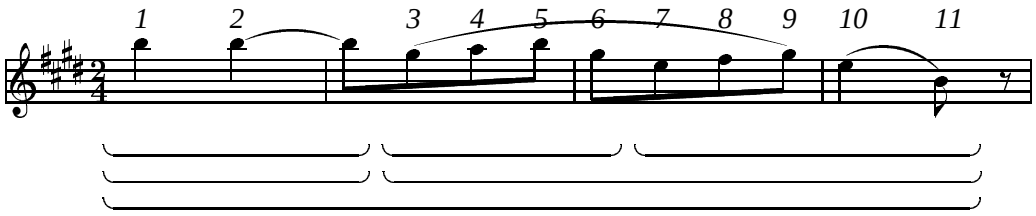


Figure 4.7: Final grouping structure for bars 3–6 of Fauré's *Berceuse*.

For this simple example, the grouping rules have correctly identified all the possible boundaries. One of the boundaries, although musically possible, was considered too insignificant to remain in the final analysis. The relative strength of the boundaries correctly builds the complete grouping hierarchy for this phrase.

4.6.2 Grouping: Excerpt from *Mozart's G Minor Symphony*

Figure 4.8 shows the results of applying the grouping rules to the opening bars of Mozart's G Minor Symphony. The points at which the preference rules can be applied are indicated beneath the score. The five rules indicated in bold denote differences between the results given by Lerdahl and Jackendoff (1983) and the automated system presented here.

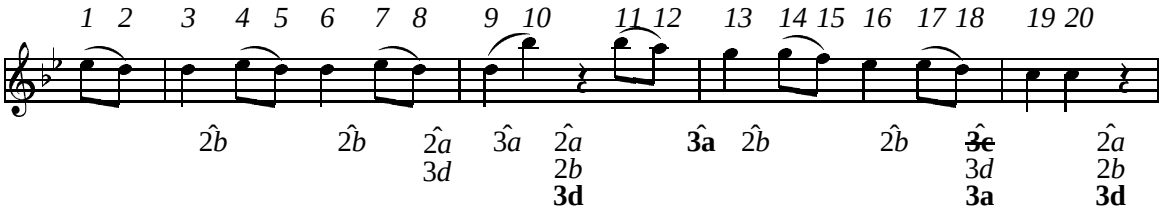


Figure 4.8: Potential grouping boundaries for the opening of Mozart's G Minor Symphony. Adapted from Lerdahl and Jackendoff (1983).

The first of these differences occurs between notes *10* and *11* where the automated application of the rules has identified an obvious instance of GPR3(d) which L&J

did not apply. The second difference, between notes *12* and *13* is more subtle. The transition from notes *11* to *12* is from B flat to A: one semitone. The transition from *13* to *14* is from G to G: zero semitones. Whereas the transition from *12* to *13* is from A to G: one whole tone, therefore GPR3(a) should be applied at this point.

There is a difference occurring between notes *18* and *19*. L&J label this boundary point with the rule GPR3(c) which, since there is no annotated change in articulation, does not apply here. However, rule GPR3(a) does apply (in an analogous situation to the previous application).

The final difference, at the end of the excerpt, corresponds to a similar situation as from *10* to *11* where an application of GPR3(d) has been missed.

There are ten possible boundary points, however it is unlikely that all of them apply in the excerpt. The grouping boundaries which are strongest are those between *10* and *11* and at the end of the excerpt. These boundaries break the excerpt into two parts of the same length with each part having a similar rhythmic structure. The boundaries also are supported by the large number of rules that apply there.

Assuming the two boundaries discussed above are active, the boundary between *9* and *10* immediately comes into question because it could potentially create groups of size 1. Upon listening to the excerpt, event *10* does stand out from the rest of the first part, but probably not strongly enough to force a grouping boundary.

The two applications of GPR2(b) that occur between *3* & *4* and *6* & *7* (and similarly for the second part of the excerpt), do form boundaries which are consistent with the excerpt. This being the case, then the boundary between *12* and *13* should be discounted because it fragments the small three event group identified in the first part and also results in the creation of a group with a single event.

We are now left with two grouping boundaries between *8* & *9* and between *18* & *19* which have yet to be decided. L&J argue that these boundary points can be ignored with the application of parallelism under GPR6. They state that if parallelism was applied and this boundary was considered valid, then the similarity between events *1–3*, *4–6* and *7–10* would cause an extra boundary to be inserted into the first two of these groups. This boundary would then create two more groups with just a single element in them and would therefore be undesirable.

A different interpretation based on the same parallelism rule would be as follows: The first two sets of events establish a pattern that consists of three events grouped together. The third set of events, although consisting of four events, begins with the same rhythmic pattern and melodic shape as the first two, and so the lack of boundary in these first two sets decreases the likelihood of a boundary in the last set.

Given the set of boundaries mentioned above, the resulting grouping hierarchy is shown in Figure 4.9. The excerpt is divided into two main parts, with each part divided into three sub-groups. This final structure is musically sensible, but again listener dependent.

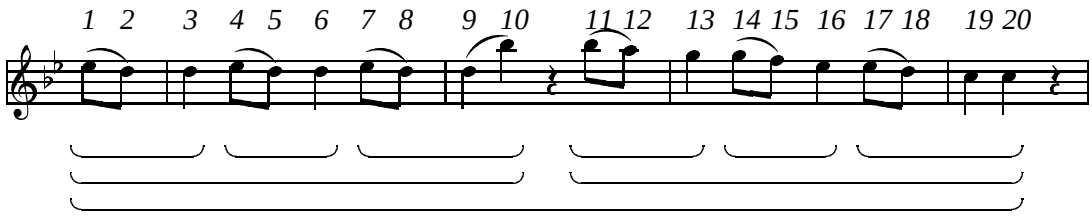


Figure 4.9: Grouping structure for the opening of Mozart's G Minor Symphony.

This second example shows the complexity that is possible even in a relatively short example. The grouping boundaries derived from the score show 2^{10} possible grouping structures. Through application of the higher-level rules and musical intuition, that set of 1024 was reduced down to one final structure. It is this ability to reduce the set of possible structures down to an individual interpretation that will be achieved by the performance analysis module.

The following three sections briefly describe the results of applying the grouping rules to three full pieces. The resulting grouping boundaries are not discussed in detail due to the over-generation inherent in GTTM.

4.6.3 Grouping: *Berceuse*

The grouping process was applied to the whole of the lead voice of *Berceuse*.[12] Figure 4.10 shows a graphical representation of the results. The height of the bars corre-

[12]Section A.1 contains the *Charm* representation for *Berceuse*; Section B.1 contains the output from the grouping component and the musical score can be found in Appendix E.

spond to the strength of the boundary rules that apply at that point i.e. the sum of their feature-category weighting factors. There are one hundred possible grouping boundaries in the analysis of *Berceuse*, which leads to a total of 2^{100} $(\approx 1.27 \times 10^{30})$ possible grouping structures.

Berceuse consists of three different sections. The first section (bars 1–34) primarily consist of a repeating four-bar phrase. The peaks near bars 7, 11, 15, etc. occur at the boundaries of these four-bar phrases. The middle section of the piece (bars 35–58), consists of a transitionary theme which is less structured than the first section. A one-bar phrase defines the last section of the piece (bars 59–83). The lead voice has swapped rôles with the second voice and performs this one-bar accompaniment until the end of the piece except for one exception (bars 74–76) when one final repetition of the four-bar phrase is performed.

4.6.4 Grouping: *Auf dem Hügel sitz ich spähend*

Figure 4.11 shows the results of applying the grouping process to the vocal part of *Auf dem Hügel sitz ich spähend*.[13] The piece consists of five sections, the boundaries of which are represented by the tall columns in bars 10, 20, 30 and 40. Each section of the piece lasts approximately nine bars and with a two-bar rest between each section. The musical structure within each section is very similar as are the detected patterns of boundaries.

There are seventy-two candidate boundary points generated by the structural analysis which results in a total number of possible hierarchies of 2^{72} $(\approx 4.72 \times 10^{21})$.

4.6.5 Grouping: *Gute Nacht*

Gute Nacht[14] consists of three sections which, as can be seen in Figure 4.12, have very similar internal structures. The sections primarily consist of a four-bar phrase with a short rest between each phrase (indicated by peaks in bars 11, 15, 19, etc.). The

[13] Section A.2 contains the *Charm* representation for *Auf dem Hügel sitz ich spähend*; Section B.2 contains the output from the grouping component and the musical score can be found in Appendix E.

[14] Section A.3 contains the *Charm* representation for *Gute Nacht*; Section B.3 contains the output from the grouping component and the musical score can be found in Appendix E.

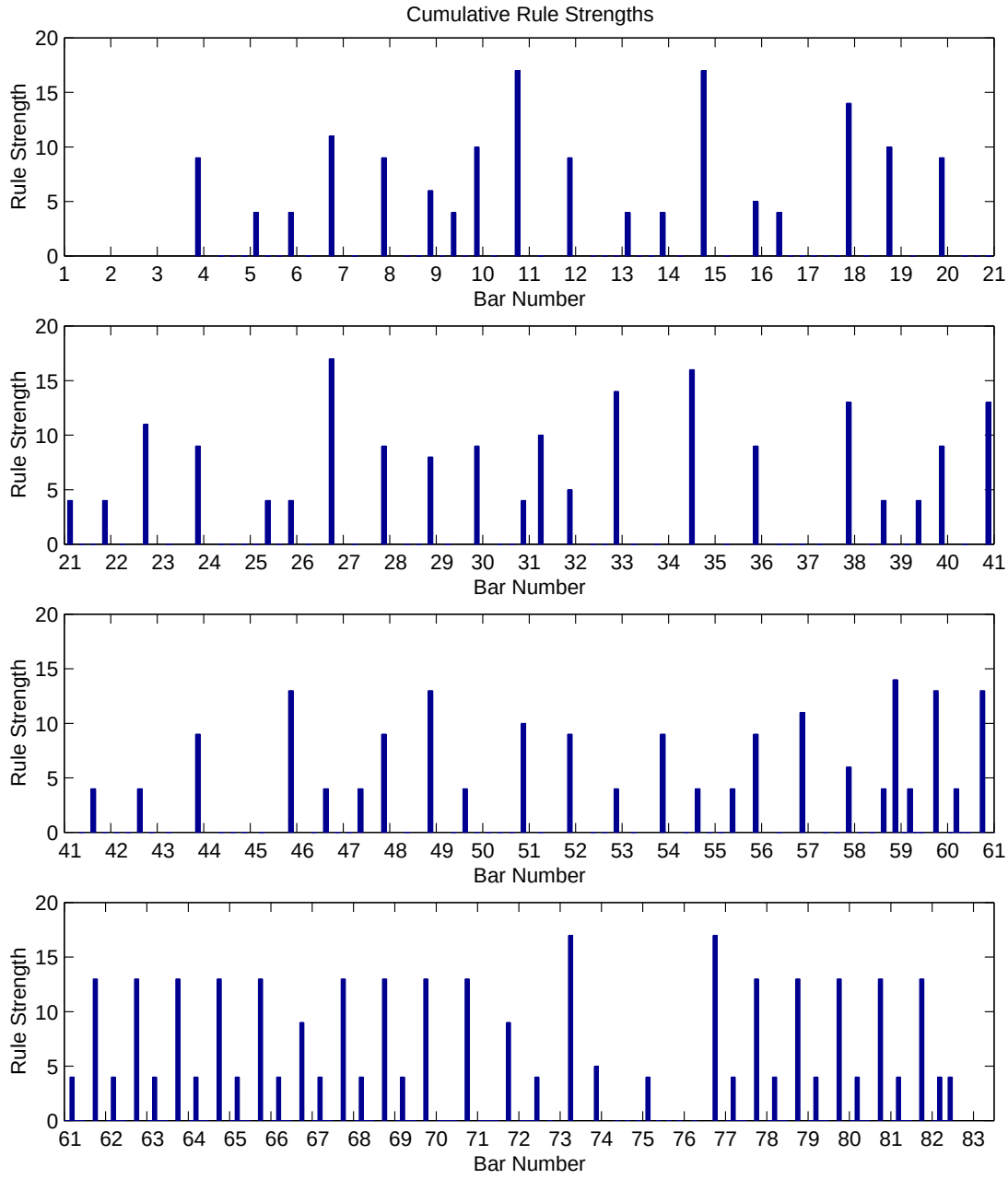


Figure 4.10: The potential grouping boundary points for *Berceuse*. Each potential boundary point is represented by a bar whose height reflects the strength of the rules that apply at that point.

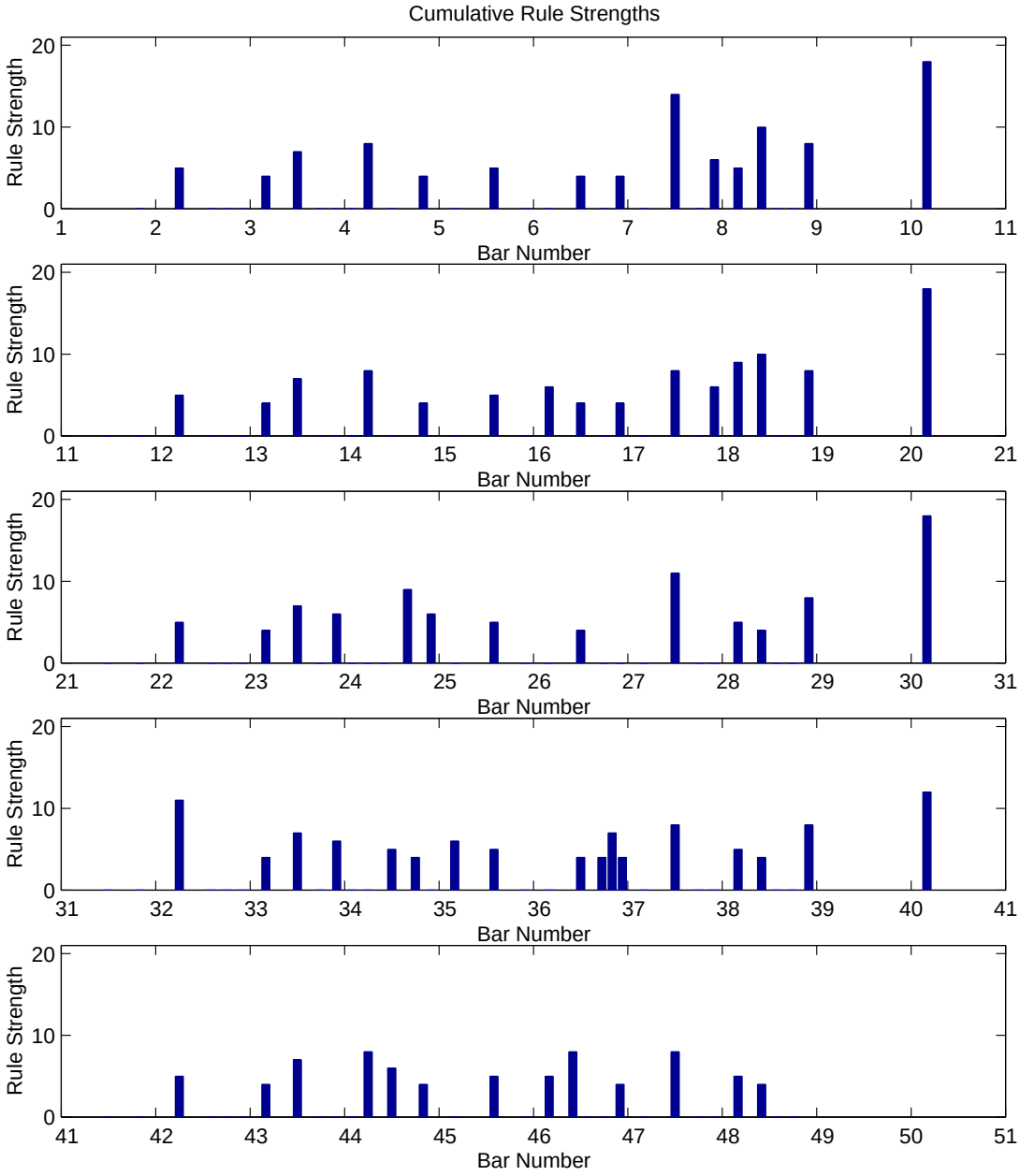


Figure 4.11: The potential grouping boundary points for *Auf dem Hügel sitz ich spähend*.

sections consist of six repetitions of this four-bar phrase with the first four repetitions separated from the last two by a two-bar rest (e.g. bars 23–25). The final section is slightly different from the first two in that it ends with an extra two-bar phrase.

The grouping analysis detects 112 possible grouping boundaries. These boundaries result in a possible number of grouping structures of $2^{112} = 5.19 \times 10^{33}$.

4.7 Summary

This chapter presented an implementation of Lerdahl and Jackendoff's Grouping Structure as a structural analysis component. It highlighted how, with the use of switches, the component supports multiple possible structures. The implementation also included the notion of the relative weighting of rules to model how some of the preference rules may be more structurally important than others.

Although the chosen weightings do generate a plausible set of grouping boundaries, the implementation does not take into account the context of a rule's application. For example, in *Gute Nacht*, the boundary in bar 11 is assigned the same total feature-category weighting as the boundary which represents the 6-bar rest between bars 33 and 39. The latter of these boundaries is much stronger and contributes significantly to the higher-level structure of the piece.

However, the overall results of the grouping rules correspond to musically sensible boundaries and, for the most part, suitable final strengths. With further experimentation into the relative strengths of the preference rules and the incorporation of context based weighting the idea of attaching weights to the grouping rules seems to be successful.

It is clear that the number of possible structures supported by the grouping preference rules is extremely large (see Table 4.6 for a summary), so an efficient representation for these structures is required.

The following chapter presents a representation which is effective, flexible and meets the requirements established here.

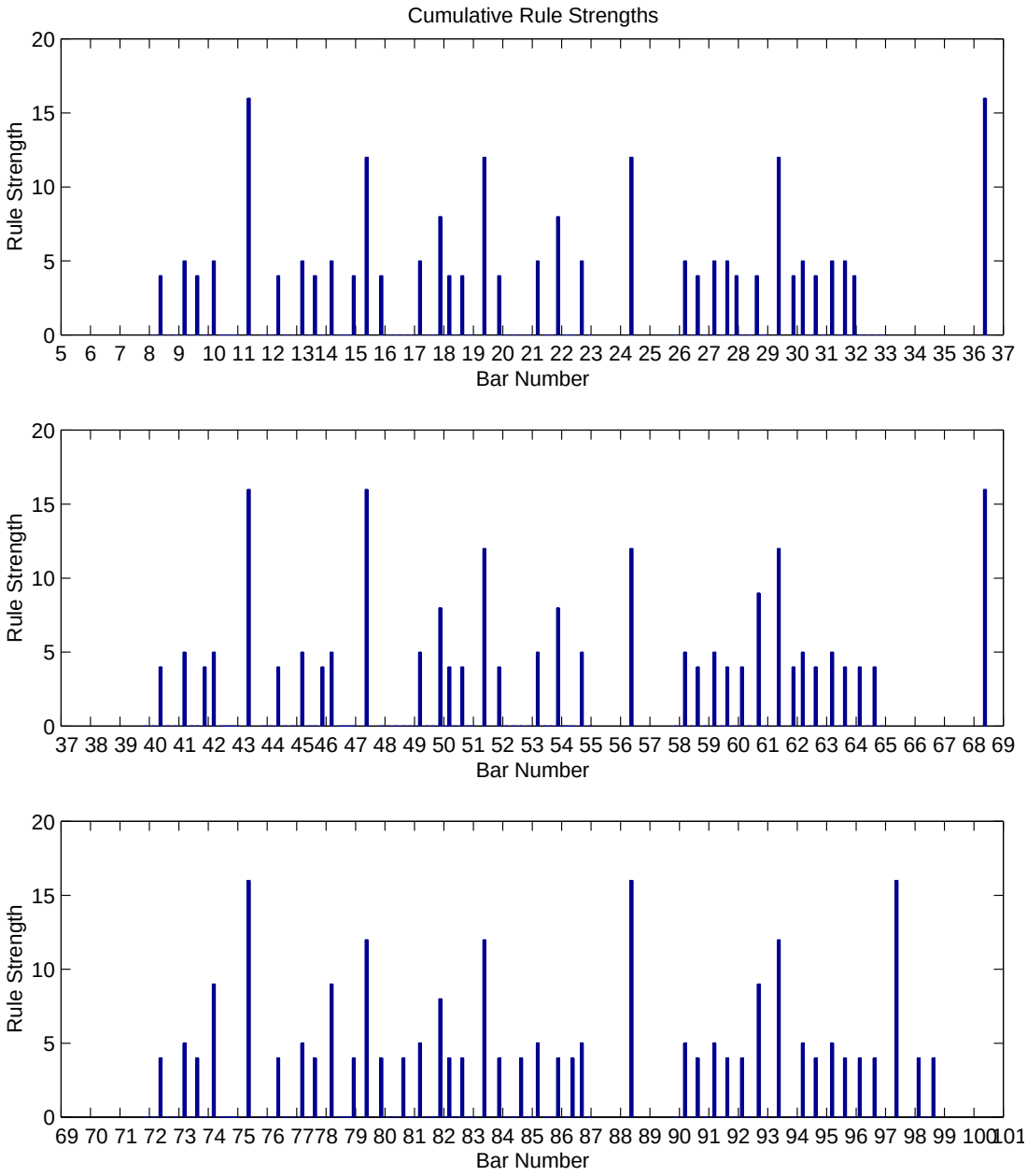


Figure 4.12: The potential grouping boundary points for *Gute Nacht*.

Piece	Boundaries	Possible Structures
Berceuse	100	$2^{100} = 1.27 \times 10^{30}$
Auf dem Hügel sitz ich spähend	72	$2^{72} = 4.72 \times 10^{21}$
Gute Nacht	112	$2^{112} = 5.19 \times 10^{33}$

Table 4.6: Table showing the number of boundaries and number of possible structures for three musical pieces.

Chapter 5

Representing Trees with Constraints

This chapter[1] describes a constraint based representation for the musical structures discovered by the analysis presented in Chapter 4.

The constraint based approach enables a large set of trees to be represented in a relatively efficient way and provides a means of allowing other parts of the GTTM analysis to interact with the grouping structure. Although the full power of this representation is not required in the final analysis process presented in this research, the representation is included as it provides a way for the individual parts of a more complete implementation of GTTM to interact.

The representation described in this chapter is intended to be used in conjunction with the results from Chapter 7 to produce the final structural analysis. The process of selecting the final analysis is discussed in detail in Chapter 9.

5.1 Introduction

The previous chapter introduced the grouping component of L&J's Generative Theory of Tonal Music. The result of applying the grouping rules to the pieces of music was a hierarchical structure that denoted which musical events belong together in groups. This hierarchical structure has a number of interesting properties:

- Every musical event belongs to a group;

[1] The majority of this chapter has been previously published as Curry et al. (2000)

- Each larger scale group consists entirely of smaller groups;
- At the coarsest structural level, the entire piece forms a group.

These group properties can be used to prescribe a particular class of tree which has the following properties:

Rooted - each tree has a node distinguished as the root node.

Ordered - the children of each node are distinct and cannot be re-ordered without changing what the tree represents.

Constant depth - the leaf nodes of each tree are all the same distance from the root.

Strict - at each depth, one of the nodes has at least two successors.

The number of distinct trees in this class is large for each n, where n is the number of leaf nodes. If $n \geq 10$ the set of trees described can not easily be manipulated or used within a computer system. Presented below is an efficient way of representing this large set of trees, using constraint logic programming, that enables this class of trees to be used in this research.

5.2 Motivation: Grouping Structure

Figure 5.1 shows an example of a grouping structure for a small excerpt of music. The music has been segmented into five different groups, one for each collection of three notes. The musical rest between the third and fourth groups induces a grouping boundary that creates two higher level groups containing the five smaller groups. These two groups are then contained within one large group at the highest level.

The grouping structure can be represented as a tree. Figure 5.2 shows such a representation (inverted, to aid comparison) for the grouping structure shown in Figure 5.1. The leaf nodes at the top of the tree correspond to the notes in the score, and the branches convey how the notes are grouped together. The aim of this part of the research is to represent the class of trees of which this is an example. From this point

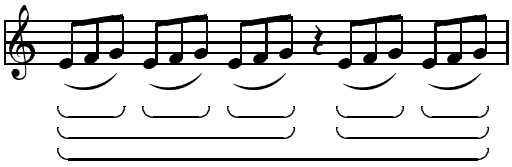


Figure 5.1: An example grouping structure

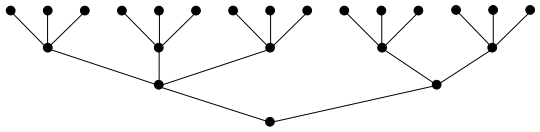


Figure 5.2: Tree representing the grouping structure shown in Figure 5.1

onwards the trees will be presented in the more traditional manner, i.e. the leaf nodes at the bottom and the root node at the top.

Figure 5.3 shows an example of another possible grouping structure for the same small excerpt of music. The music has been segmented into four different groups, one for each collection of three notes before the rest, and then the remaining events have all been grouped into one group. In this case, the grouping structure hierarchy is unbalanced in the sense that the events on the left hand side of the rest have a hierarchical depth of three groups whereas the events on the right have a depth of two groups.

Figure 5.4 shows such a tree representation for the grouping structure shown in Figure 5.3. Importantly, unlike the previous example, one of the nodes of this tree only has one branch extending from it. This special case occurs whenever the hierarchy is unbalanced in terms of the grouping structure and is intended to convey the fact that no new hierarchical information related to the grouping structure is contained in that particular node. Importantly, at the lowest level of the hierarchy it is a valid grouping concept to have a node with a single branch extending from it to represent a singleton event.

Figure 5.5 shows an incorrect alternative tree which may have been considered for the structure in Figure 5.3. The tree in this case would represent the grouping structure shown in Figure 5.6.

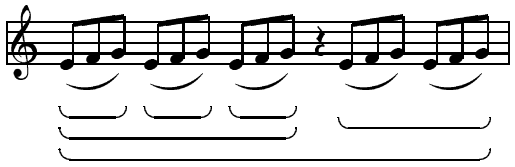


Figure 5.3: An example of grouping structure with varying hierarchical depth.

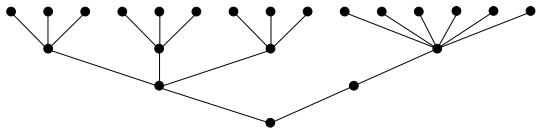


Figure 5.4: Tree representing the grouping structure shown in Figure 5.3.

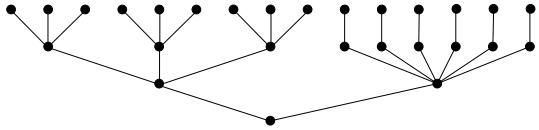


Figure 5.5: Tree which does not reflect the grouping structure shown in Figure 5.3.

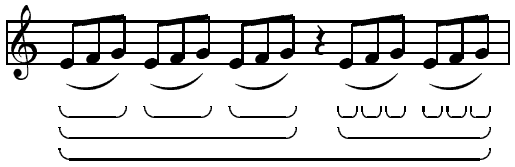


Figure 5.6: The incorrect grouping structure which would be represented by Figure 5.5.

Although the GTTM grouping rules are presented formally, the preference rules introduce a large amount of ambiguity. For a particular piece of music, there are many possible grouping structures which satisfy the preference rules. The purpose of this research is to devise a way to represent this large set of possible structures efficiently so that they can be used in the proposed computer system.

5.3 Using Constraints

This section explains how constraint logic programming (Henternryck, 1989) is used to represent sets of trees. Although constraints have been used in the areas of music composition (e.g. Henz et al. (1996)) and tree drawing (e.g. Tsuchida et al. (1997)) before, the current research concerns itself with a different problem, namely the efficient representation of large numbers of tree structures.

Constraint logic programming over finite domains enables the specification of a problem in terms of variables with a range of possible values (known as the *domain* of the variable) and equations that specify the relationships between the variables. The following examples illustrate the usage and subtlety of using constraint logic programming. If the constraints shown as (5.1)–(5.3) hold true, then domains of the variables x and y can be narrowed as shown in (5.4):

$$x \in \{1 \ldots 4\} \tag{5.1}$$

$$y \in \{3 \ldots 6\} \tag{5.2}$$

$$x + y \geq 9 \tag{5.3}$$

$$x \in \{3 \ldots 4\} \wedge y \in \{5 \ldots 6\} \tag{5.4}$$

Another example, borrowed from Carlsson et al. (1997), illustrates why care must be taken when using constraints. Given the constraints applied to x, y and t shown in (5.5)–(5.7), the solution is (5.8). Note the domains of y and t.

$$x \in \{1 \ldots 5\} \tag{5.5}$$

$$y \in \{2 \ldots 8\} \tag{5.6}$$

$$x + y = t \tag{5.7}$$

$$x \in \{1 \ldots 5\} \wedge y \in \{2 \ldots 8\} \wedge t \in \{3 \ldots 13\} \tag{5.8}$$

Now, if the statements of the constraints are rearranged, so that the same domains for x and t as resulted from the constraints above are posted, and the relation between x, y and t is then applied to these variables (see (5.9)–(5.11), it can be seen that the domain of y is not $\{2\ldots8\}$ as expected but $\{\text{-}2\ldots12\}$ (see (5.12)).

$$x \in \{1\ldots5\} \tag{5.9}$$

$$t \in \{3\ldots13\} \tag{5.10}$$

$$x + y = t \tag{5.11}$$

$$x \in \{1\ldots5\} \wedge t \in \{3\ldots13\} \wedge y \in \{-2\ldots12\} \tag{5.12}$$

The next sections discuss the representation of the tree nodes and presents the five types of constraints used to ensure that the generated trees belong to the desired class.

5.3.1 Representation

The class of trees will be monotonically decreasing in width from the leaf nodes up to the root and, therefore, they can be represented by a triangular point lattice of nodes which decreases in width as we progress from one level to the level above.[2] Figure 5.7 shows the point lattices for trees of width $n = 3$ and $n = 4$.

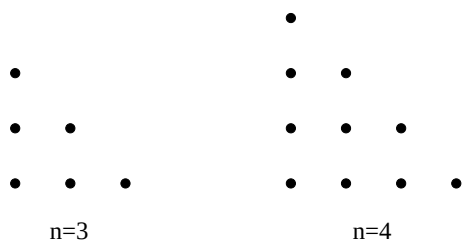


Figure 5.7: Point lattices for trees of width 3 and 4

Each node has the following variables (illustrated in Figure 5.8):

1. *id*: a unique identifier;

[2] An implementation detail means that there is always a path from the highest node of the point lattice to the leaf nodes, but this highest node should not be considered the root node. The root node may occur at any height in the point lattice and is identified as the highest node with more than one child.

2. *uplink*: a connection to the level above;

3. Downlink values which represent all the nodes on the level below that are connected to this one.

The *id* is specified as a pair of (x,y) coordinates to simplify the implementation details. The *uplink* variable contains an integer that represents the *x*-coordinate of the node on the level above to which the current node is connected i.e. node (*uplink*, $y+1$). The downlink values, specified by a lower (*dl*) and upper (*du*) bound, refer to a contiguous range of nodes on the level below that may be connected to the current node i.e. nodes (*dl*, $y-1$)...(*du*, $y-1$).

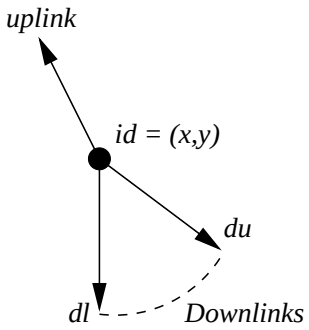


Figure 5.8: A typical node

The next sections present the constraints that are applied to the nodes in order to create the specific set of trees which are of interest. They begin by specifying the domains of the variables and then constraining the nodes so that only those trees that belong to the desired class can be generated.

5.3.2 Node Constraints

The first task is to define the domains of the variables for each node. Due to the triangular shape of the point lattice, the uplink for each node is constrained to point either directly upwards, or up and to the left of the current node. The downlink for each node is similarly constrained to span the nodes directly below, and below and to the right of the current node.

The constraints (given in (5.13)-(5.16)) define the domains of the *uplink* and downlink range (i.e. *dl* and *du*) for each node.[3] The *uplink* lies in the range $\{0 \ldots x\}$ where x is the x-coordinate of the current node. The zero in the range is used when the node is not connected to the level above.

$$domain([uplink]) = \{0 \ldots x\} \tag{5.13}$$

$$domain([dl, du]) = \{0 \ldots n\} \tag{5.14}$$

$$(dl = 0) \oplus (dl \geq x) \tag{5.15}$$

$$du \geq dl \tag{5.16}$$

The downlink specifiers *dl* and *du* are constrained in a similar way to lie in a range from $\{0 \ldots \mathrm{n}\}$ with the added constraints that *du* has to be greater than or equal to *dl* and that *dl* either equals zero or is greater than or equal to x. Figure 5.9 shows how these constraints relate to the direction of the connections to and from each node.

Constraint (5.17) handles the situation where a node is not used in a tree. If the *uplink* of the node is zero then the downlinks of the node must also be zero.

$$((dl = 0) \Leftrightarrow (du = 0)) \wedge ((dl = 0) \Leftrightarrow (uplink = 0)) \tag{5.17}$$

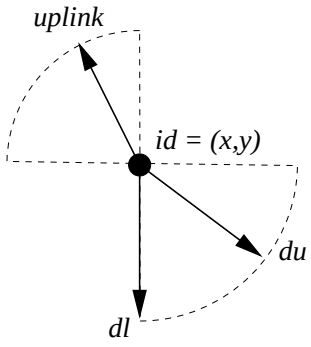


Figure 5.9: Constraining the Uplinks and Downlinks

[3] The $\oplus$ in (5.15) denotes exclusive-or.

5.3.3 Level Constraints

To ensure that the connections between two levels do not cross, constraints (5.18) and (5.19) are applied to each pair of adjacent nodes on a level. For a pair of nodes A and B, with A directly to the left of B, the $uplink_B$ must either point to the same node as the $uplink_A$ or to the node immediately to the right of it or, if it is unused, be equal to zero (5.18).

$$(uplink_B = uplink_A) \vee (uplink_B = uplink_A + 1) \vee (uplink_B = 0) \tag{5.18}$$

Once one of the uplinks on a particular level becomes equal to zero, all the uplinks to the right of it must also be zero (5.19). This prevents the situation of an unconnected node in the midst of connected ones.

$$(uplink_A = 0) \Rightarrow (uplink_B = 0) \tag{5.19}$$

Figure 5.10 shows examples of correct and incorrect mid-sections of a tree under these new constraints. The bottom example is incorrect because it violates constraints (5.18) and (5.19).

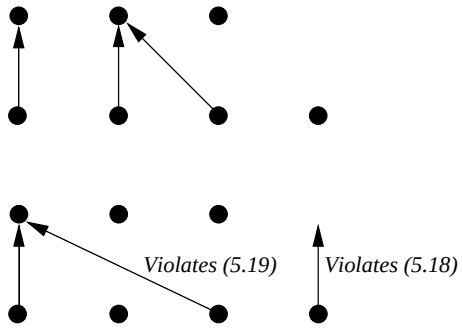


Figure 5.10: A correct (*top*) and incorrect (*bottom*) mid-section of a tree

5.3.4 Consistency Constraints

If the current node (B) refers to a node (A) in the level above, the x-coordinate of B must occur within the downlink range of A. Constraint (5.20) ensures that if this node

(B) points to a node (A) on the level above, the downlink range of A must include B. Figure 5.11 shows how this constraint affects two nodes where the lower one is connected to the upper one.

$$(x_A = uplink_B) \Leftrightarrow ((x_B \geq dl_A) \wedge (x_B \leq du_A)) \tag{5.20}$$

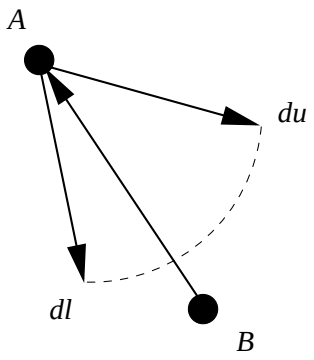


Figure 5.11: Ensuring connectivity between nodes on different levels

5.3.5 Width Constraints

The trees are now constrained to decrease in width, level by level, from the leaf nodes to the root node. The width of a level is defined as the number of nodes that have a non-zero uplink on that level. Constraint (5.21) deals with this situation with the precondition that the width of the current level is greater than 1. This precondition is necessary to allow situations such as the first four trees in Figure 5.13, in which the root node is considered to be at the point where branching begins.

$$(width_i > 1) \Rightarrow (width_j < width_i) \tag{5.21}$$

We want to ensure that the trees decrease in width to reduce the search space as much as possible. Figure 5.12 shows an example of a tree which does not decrease in width between two levels, this tree can be removed from the search space as it does not contribute any new information to the grouping structure between levels i and j.

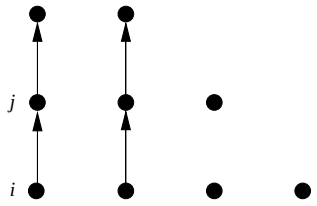


Figure 5.12: A section of a tree that does not decrease in width

5.3.6 Edge Constraints

The last step is to ensure that the uplink of the rightmost node on a level (i) points inwards (the rightmost node in the top diagram of Figure 5.10 is an example of this).[4] The maximum x of the level above (j) that has a non-zero *uplink* is found, and then the *uplink* of the rightmost node on this level (i) is constrained to point to it ((5.22) and (5.23)).

$$\mathcal{S} \equiv \{x \mid id(x,y) \in \{level_j\} \wedge uplink_{id(x,y)} \neq 0\} \tag{5.22}$$

$$\forall A \,.\, A \in \{level_i\} \Rightarrow uplink_A \leq \max(\mathcal{S}) \tag{5.23}$$

5.3.7 Valid Trees/Grouping Structures

The constraints given in Sections 5.3.2 to 5.3.6 define the set of trees which belong to the desired class. Figure 5.13 shows an example set of width $n = 4$. The white nodes are ones that occur in the generated solutions but are not considered to be part of the tree since the root of the tree is the highest node with more than one child.

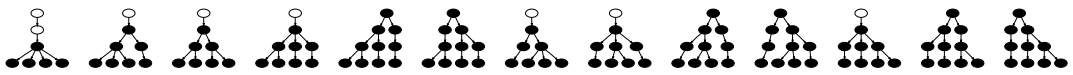


Figure 5.13: All the trees of width four ($n = 4$)

If each of the leaf nodes of the trees is viewed as representing a musical event, then

[4] 'Rightmost' means the node on the current level with the maximum x-coordinate that has a non-zero uplink.

each tree represents a potential grouping structure. Importantly, all the trees in the generated set conform to the grouping well-formedness rules discussed in Section 4.3.1.

5.3.8 Using the Constraint Representation

The constraints which have been defined in the sections above describe a general class of trees. The next step is to introduce aspects of the grouping structure that reduce this large set of trees to a subset that corresponds to the piece of music being analysed.

As discussed in the previous chapter, every point in the musical score where a grouping boundary could occur is identified and, for each of these points, the relative strength of this boundary against the surrounding ones is calculated. Each of these boundary points can then be used to determine the shape of the tree by controlling the degree of separation of the leaf nodes which represent the relevant musical events. For example, if two musical events are separated by a grouping boundary of strength one, the leaf nodes which represent these events can be forced to have different parents.

To separate the nodes in a tree, the parents of the nodes must not be the same, and, if there is a measure of relative strength between boundaries, it can be used to specify how far towards the root the nodes need to be separated (see Algorithm 5.1).

Algorithm 5.1 Recursive algorithm to repel nodes to a height *strength*.

REPEL$(id_A, id_B, strength)$

1: **if** $strength \geq 1$ **then**
2: PARENT$(id_A) \neq$PARENT(id_B)
3: REPEL(PARENT(id_A),PARENT$(id_B), strength - 1)$
4: **end if**

This recursive predicate takes two nodes and a strength argument and recursively ensures that the nodes are separated up to a height *strength*. Figure 5.14 shows an example tree where the tree is divided into two subtrees by a REPEL constraint that is applied with $strength = 1$ between the second and third leaf nodes.

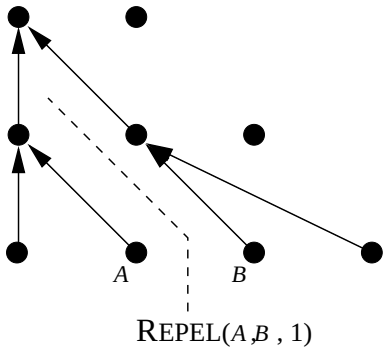


Figure 5.14: How REPEL affects the tree

5.4 Results

We generated all the trees up to width $n = 7$ and found a similarity with an entry in the Online Encyclopedia of Integer Sequences (Sloane, 2000). The number of trees for a given width matched a sequence discovered by the mathematician Arthur Cayley (1891) based upon this particular class of trees which has the recurrence shown in (5.24) and (5.25)[5]

This recurrence defines the number of trees that belong to the desired class that are of width n.

$$a(0) = 1 \tag{5.24}$$

$$a(n) = \sum_{k=1}^{n} \binom{n}{k} a(n-k) \tag{5.25}$$

Using the representation proposed here, the approximate formula, derived experimentally, for the number of constraints to represent the set of all the trees of width n is given in (5.26).

$$\text{Constraints} \approx \frac{2}{3}n^3 + 11n^2 - \frac{2}{3}n - 24 \tag{5.26}$$

The number of trees of width n grows rapidly (e.g. the number of trees of width 50 is 1.995×10^{72}). By contrast, the number of constraints it takes to represent the same number of trees is 1.1×10^5.

[5]Where $\binom{n}{k}$ is the standard n choose k formula given by: $\frac{n!}{k!(n-k)!}$

Figure 5.15 shows how the number of trees grows in comparison to the number of constraints as the width of the tree increases. As can be seen, the number of trees increases at a higher than exponential rate whereas the number of constraints increases at a low-order polynomial rate.

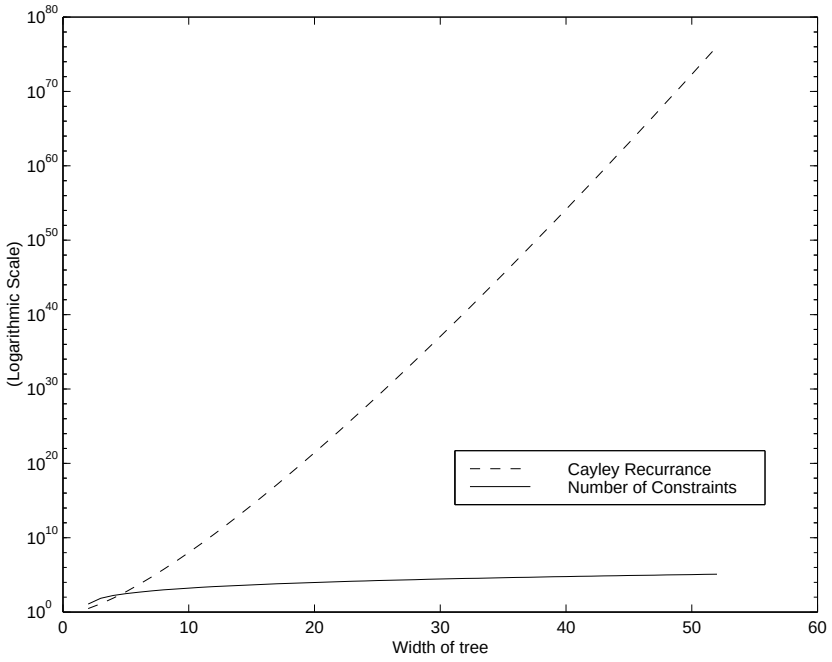


Figure 5.15: A graph showing how the number of trees and number of constraints grow with the width of the tree

5.5 Summary

This chapter presents research on representing a specific class of trees with constraint logic programming. Although the number of constraints needed to represent these large sets of trees is comparatively small, the computational time needed to solve the constraints is not.

The representation currently restricts the trees to have leaf nodes at the same depth; however, it does allow the addition of quite simple constraints to change the class of trees represented. For example, to restrict the trees to strictly binary trees the further

constraint $du = dl + 1$ need only be applied.

Through the use of constraints, the generation of trees has been delayed until all the possible restrictions has been added. This offers a great reduction in complexity and allows trees of greater width than would normally be possible to be manipulated.

Chapter 6

Empirical Study

6.1 Introduction

This chapter describes a study conducted to test the feasibility of the principles on which the performance analysis module is based and to gather data for subsequent use in the system.

The study consisted of two phases which were conducted to corroborate Repp's (1996) conclusions and to collect performance data of a number of pieces by different musicians. Two pairs of musicians contributed to this study by performing, in total, three different duets five times, giving a total of fifteen recorded performances.

Chapter 7 describes how the data recorded from this study was analysed to identify features salient to resolving the ambiguity of the structural analysis.

6.2 Aims

The two principal aims of this study were:

1. Corroborate Repp's conclusion that performers tend to be consistent, with regards to their expression, across performances.

2. Gather performance data for subsequent analysis.

Aim 1 is important because if there is no consistency between performances by the same performer, then that suggests that there is not a pattern in the performance which can be used to resolve the structural ambiguity of the piece. For the purpose of this study, the expressive timing of the performances will be closely studied rather than other features of the performance such as dynamics or timbre.

Provided Aim 1 is achieved then the achievement of Aim 2 will result in a set of data which can be used to test the structural resolution features of the system.

6.3 Objectives

The two objectives that correspond to the aims given above are:

A Demonstrate that there is a significant similarity in the expressive timing between a number of performances of the same piece by the same performers.

B Gather a number of MIDI files which represent several performances of a number of pieces.

6.4 Using MIDI as a medium for recording

Before continuing with the description of the empirical study it is important to mention why MIDI was chosen and why it provides a suitable medium for recording data. MIDI (Rothstein, 1992) was designed to allow MIDI enabled instruments such as synthesisers to communicate with one another. MIDI has also become a standard means of storing musical data. A MIDI file consists of a number of instructions that describe the musical events and effects that should be performed by the device playing that MIDI file.

The basic units of information in the MIDI format are the ‘Note On’ and ‘Note Off’ MIDI events. The ‘Note On’ signal passes a pitch and a velocity value to the MIDI device which starts playing that note with a corresponding loudness. Every ‘Note On’ signal needs a corresponding ‘Note Off’ signal; this signal also has a pitch and a velocity which instructs the MIDI device to stop playing the corresponding note.

The ‘Note On’ and ‘Note Off’ signals correspond to a key press and a key release of a keyboard. The velocity represents the speed at which the key was pressed (for ‘Note On’) and released (for ‘Note Off’).

There are many other signals defined as part of the MIDI specification. These include signals to express key signature, tempo and channel information.[1] Because of the relatively small amount of information stored in the MIDI file, when compared with files which store an audio signal, they are a popular means of storing performances. However, the format does not allow an easy way of specifying the timbre of the output sounds and it is often trivial for a listener to distinguish between a MIDI generated performance and a recording of a performance on ‘real’ instruments. This study concentrates on the timing of the events, so the lack of timbre is not an issue; the ‘Note On’ events are the ones which are of most interest.

Given the large amount of performance data (more than 10,000 musical events) and the possible problems of disambiguating the output of each piano from an audio signal, MIDI is an ideal recording medium.

6.5 Phase I

The goal of this first phase of the study was principally to achieve objective A.

6.5.1 Participants

Two musicians contributed to this phase of the experiment:

Musician A: a lecturer at the University of Edinburgh who is an accomplished accompanist.

Musician B: a PhD candidate at the University of Edinburgh who is a self-taught classical/popular musician who is a proficient amateur and used to group performance.

[1] Often different MIDI devices are assigned different channels, this allows one stream of MIDI data to control many different devices or instruments.

Both musicians gave their time freely and had practised the piece individually before the date of recording.

6.5.2 Music

For this phase of the study, *Berceuse* from Gabriel Fauré's "Dolly Suite" (Op. 56) was the piece chosen to be performed. The piece is a piano duet and was written between 1893 and 1896 in the Romantic style.

Although the lead voice is not strictly monophonic, as required for GTTM analysis, the vast majority of the musical events consist of pairs of notes which are an octave apart and so for the purposes of this study can be treated as monophonic.

The piece consists of a four-bar repeating phrase played in the first instance by the lead voice, with the second voice providing a repeating, one-bar phrase as the accompaniment. After eight repetitions of the four-bar phrase (bars 3–34), the lead voice embarks on a melodic answer during the central section of the piece (bars 35–58). The lead voice then swaps rôles with the second voice until the end of the piece (apart from one more repetition of the original four-bar phrase (bars 73–76)).

6.5.3 Equipment

Two Yamaha Clavinova CLP-360 keyboards with realistic weighted action and MIDI output were used to record the performances. The keyboards were both connected to a Macintosh Quadra 650 computer recording the MIDI outputs. Each piano was assigned a different MIDI channel to distinguish between the musical events performed by the two musicians.

The MIDI data of the performances was recorded using Opcode's MAX (Dobrain, 1999) software running on the Macintosh. MAX is a graphically based object oriented programming environment which allows the development of programs using predefined objects that can be connected together using the MAX graphical interface. The program developed for the purposes of these experiments, see Figure 6.1, took the MIDI inputs from the two pianos and stored the data representing the musicians' key presses into a MIDI format file.

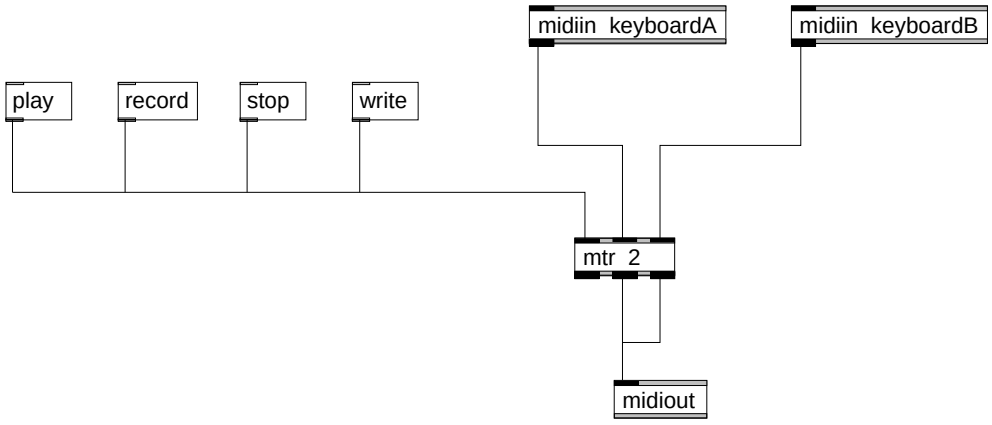


Figure 6.1: Graphical representation of the MAX program.

It was necessary to manipulate the recorded MIDI files to clean up the data in ways described below. The Rosegarden package (Green et al., 2002) running under Solaris was used to perform this task. Rosegarden is a free software suite which includes a sequencer and a notational editor. The MIDI performances were later exported into a textual representation by Rosegarden for input to the rest of the system.

The study took place in a private room in the Faculty of Music. The only people present in the room were the two musicians performing the piece and the observer.[2] The observer had previously arranged the room so that all the equipment was prepared and ready to record as shown in Figure 6.2. The musicians were placed so that they were able to communicate, both visually and aurally, with one another (see p. 25, Section 2.5).

6.5.4 Procedure

Before recording was started, the two musicians took approximately thirty minutes to practise the piece, become comfortable with the instruments and to decide which rôles they wanted to perform.

For the recorded performances, musician A took the rôle of the lead voice and musician B that of the accompaniment.

[2]The experimental observer was the author.

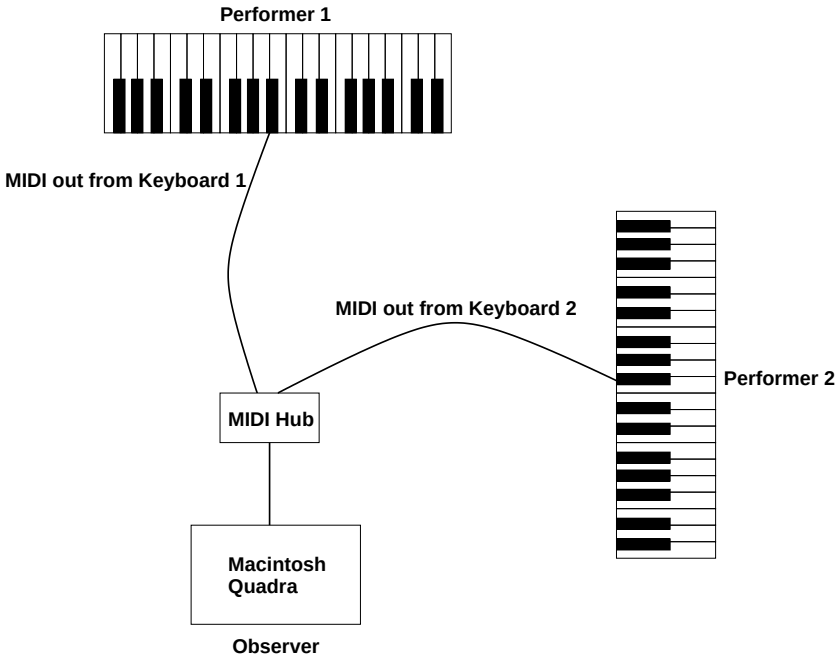


Figure 6.2: Diagram showing how the experimental equipment was arranged.

A total of ten recordings were made. The recordings were spread over three recording sessions with breaks between them. After the first three recordings were made in the first session, the performers were asked to pause for a while and listen to some unrelated music before continuing with the experiment. After this break, another three recordings were made before another break during which another different piece of music was played to the musicians. Finally, during the last session, the musicians recorded another four performances of the piece.

The purpose of introducing breaks between performances was to simulate the rehearsal process when a piece may be over a number of weeks with long pauses between sets of rehearsal performances. Due to time and equipment constraints, it was not practical to spread the study over a number of weeks so the breaks between performances were introduced to simulate, to some extent, this effect. Playing different pieces of music to the performers during the break was used to try to enhance this effect.

6.5.5 Results

Rosegarden was used to inspect the MIDI files containing the performance recordings and to identify any problems such as performance errors. As one of the simplifying assumptions of the model is that the performances are note perfect, any performance errors had to be corrected as described below.

6.5.5.1 Data Manipulation

There were three different errors that occurred in the recordings:

Translated Notes e.g. the performer plays a C instead of a D;

Missing Notes e.g. a note that occurs in the score that does not occur in the performance;

Extra Notes e.g. a note that occurs in the performance that does not occur in the score.

Locating the errors was not difficult, however identifying the category of each error can be non-trivial.[3] For example, the simultaneous occurrence of a missing note and an extra note can be misconstrued as one instance of a translated note. However, in most cases the category of the error was readily identified and, in the more difficult cases, examination of the surrounding material and of the dynamic intensity of the notes provided enough information to resolve the error category.

Once the categories of the performance errors were identified, they were corrected as follows:

Translated Note The note pitch was adjusted to match the pitch prescribed by the score. All other aspects of the note remained the same.

Missing Note An extra note was inserted into the MIDI file to match the note prescribed by the score. The time at which the note should occur in the performance was determined by examining the timing of the missing note in other

[3] The translated, missing and extra note categories correspond to substitution, deletion and insertion errors in Desain et al. (1997).

performances and placing it at a point equal to the average ratio of the note position in the other performances. This meant that the addition of the note did not affect the overall average timing for that note within the study.

Extra Notes The note was simply removed from the performance.

Treating these errors as isolated events does not take into account the possible effects they could have on neighbouring areas of the piece. For example, if a performer has accidentally struck two keys instead of the one they were aiming for, that may alter the properties of the key strike for the note that is of interest.

However, for the purposes of this study it is assumed that it is sufficient to deal with errors as described above and that they will not have a significant effect on the remainder of the performance of the piece.

Of the ten performances of the Fauré piece, two were deemed unsuitable for further use due to the large number of errors present. The errors were not resolvable using the methods described because they occurred in clusters of adjacent events which prevented the use of an informed error correction strategy.

From the remaining eight performances, the five performances with the fewest errors were selected. It is on these performances that the following analysis is conducted.

6.5.5.2 Consistency Analysis - General Considerations

As discussed in Chapter 2, Repp (1995, 1997b) shows that both across performers and across performances there is a striking amount of similarity in timing profiles.

In order to obtain information which will allow the disambiguation of the musical structure of the piece, the performances of the piece by the musicians will have to show this consistency. If the performances do not show patterns of consistency, it would suggest that the structural information is not conveyed through the timing profiles.

Three aspects of the timing profile are readily examinable for each piece. They are:

1. Total performance duration;
2. Note duration cross-correlation across performances;
3. Note duration variance.

The ‘Total Performance Duration’ is simply a measure of the total duration of the performance of a piece. Provided the total durations of the performances are similar, the performances can all be scaled to have the same overall duration without significantly affecting the expressive timing of the performances (Desain and Honing, 1992).

The ‘Note Duration Cross-correlation’ is a more detailed examination of the duration of each event, measured in terms of IOIs with respect to the same event in each of the other performances. If there is a strong correlation between all pairs of performances, then the desired consistency exists.

The final measure, ‘Note Duration Variance’, examines the variance in duration for each event across the five performances. This measure is used to identify which events contribute significantly to any tests of overall similarity and explore why this is the case.

6.5.5.3 Consistency Analysis of *Berceuse*

The resulting MIDI files were exported to text format with the aid of Rosegarden. The textual output contains a line for every MIDI event. The first step was to filter the events to leave the ‘Note On’ events only. The content of the file then consists of a list of ‘Note On’ events in the following format:

```
<Event Time> <Event Name> [<Event Data>]
```

where `<Event Time>` is specified in terms of the number of beats from the start of the recording.

Figure 6.3 shows an example of this textual output. To obtain the time of the events in seconds, the following formula must be used, where *Timebase* is 96 and the *Microseconds*/*tick* are 5208.3 (for 120 BPM):[4]

$$\begin{aligned} \text{Time in Seconds} &= \frac{(\mathit{Timebase} \times \mathit{Beat}) \times (\mathit{Microseconds}/\mathit{tick})}{1,000,000} \\ &= \frac{(96 \times \mathit{Beat}) \times (5208.3)}{1,000,000} \\ &= \mathit{Beat} \times 0.4999968 \end{aligned} \tag{6.1}$$

[4]These values are MIDI device/file-format dependent.

```
39.36:  Note On:  1  F  6  59  1.47
40.69:  Note On:  1  E  6  61  1.29
41.95:  Note On:  1  D  6  62  1.46
43.21:  Note On:  1  A  5  51  1.34
44.45:  Note On:  1  F  5  52  1.28
45.72:  Note On:  1  E  5  61  1.99
```

Figure 6.3: Excerpt from textual representation of a performance showing the times and properties of some MIDI 'Note On' events.

Applying Equation 6.1 to the event times listed in Figure 6.3 will give the time in seconds at which those events were performed. For example, the first event listed occurred 19.68 seconds after recording began.

Using the above equation, the total durations for the performances can be calculated. Table 6.1 shows the duration times for the five performances of *Berceuse*. The arithmetic mean performance time for the piece was 197.40 seconds with a maximum deviation of 1.83% (or 3.62 seconds) from this mean.

Performance	Start Time	End Time	Duration	% of Average	Difference
1	7.38	208.40	201.02	101.83%	+1.83%
2	9.31	206.78	197.47	100.04%	+0.04%
3	7.80	205.39	197.59	100.10%	+0.10%
4	8.17	203.42	195.25	98.91%	-1.09%
5	6.17	201.85	195.68	99.13%	-0.87%
	Arithmetic Mean $\mu = 197.40$				
	Standard Deviation $\sigma = 2.04$				

Table 6.1: Durations of the five *Berceuse* performances in seconds.

Because the durations of these performances are so similar, it is possible to scale them to the same nominal length. This means that all the expressive timings are scaled by a common factor within each performance. If the performances had significantly different durations, it would not be musically sensible to perform this normalisation

because expressive timing cannot be linearly scaled as discussed by Desain and Honing (1992).

Figure 6.4 is a graph showing a timing profile of the five scaled performances. The x-axis of the graph is labelled according to the bar number. The y-axis shows the duration of each event based on IOIs i.e. the time from the start of the current event to the beginning of the next.[5]

The most striking aspect of the graph is the visual similarity of the five performances. It is not initially obvious that there are five different lines plotted on the graph.

The first occurrence of the four-bar repeating phrase, discussed in Section 6.5.2, can be seen occurring between bars 3 and 7 on the graph. This same timing structure can clearly be seen repeating in the next 20 bars.

In bar 35, an answering melody begins which lasts for the next 24 bars. The melody has a less obvious musical structure from the first section and leads the listener smoothly to the last third of the piece.

At bar 59, the lead and accompaniment voices swap rôles, with the accompanist performing the original melody and the lead voice providing the accompaniment. This continues until the end of the piece apart from a brief exchange during bars 73–78.

Figure 6.5 shows a graph of the variance of the IOIs for each event across the five performances. The two lines on the graph show the same data but scaled to different degrees. The solid line shows the variance up to 1×10^{-6}; the dashed line shows variance up to 1×10^{-7} (in order to show some of the lower-level detail).

The first three small peaks (near bars 7, 11 and 15) coincide with the end of each occurrence of the phrase. The larger peak at bar 18 occurs at the end of a crescendo marked in the score. The peaks in bars 22 and 26 also correspond to the end of phrases. The peak in bar 26 is not readily explained from the score and closer examination of the graph of performances at that point shows a relatively large spread of values.

The next two peaks in bars 29 and 33 mark the start and end of a *swell* marked in the score. The high variance from bar 35 to 37 occurs when the performers are first

[5]The durations of all the pieces were scaled to last one unit of time, to facilitate cross performance analysis, therefore the duration of each event is measured in terms of this single unit and are recorded as small real numbers.

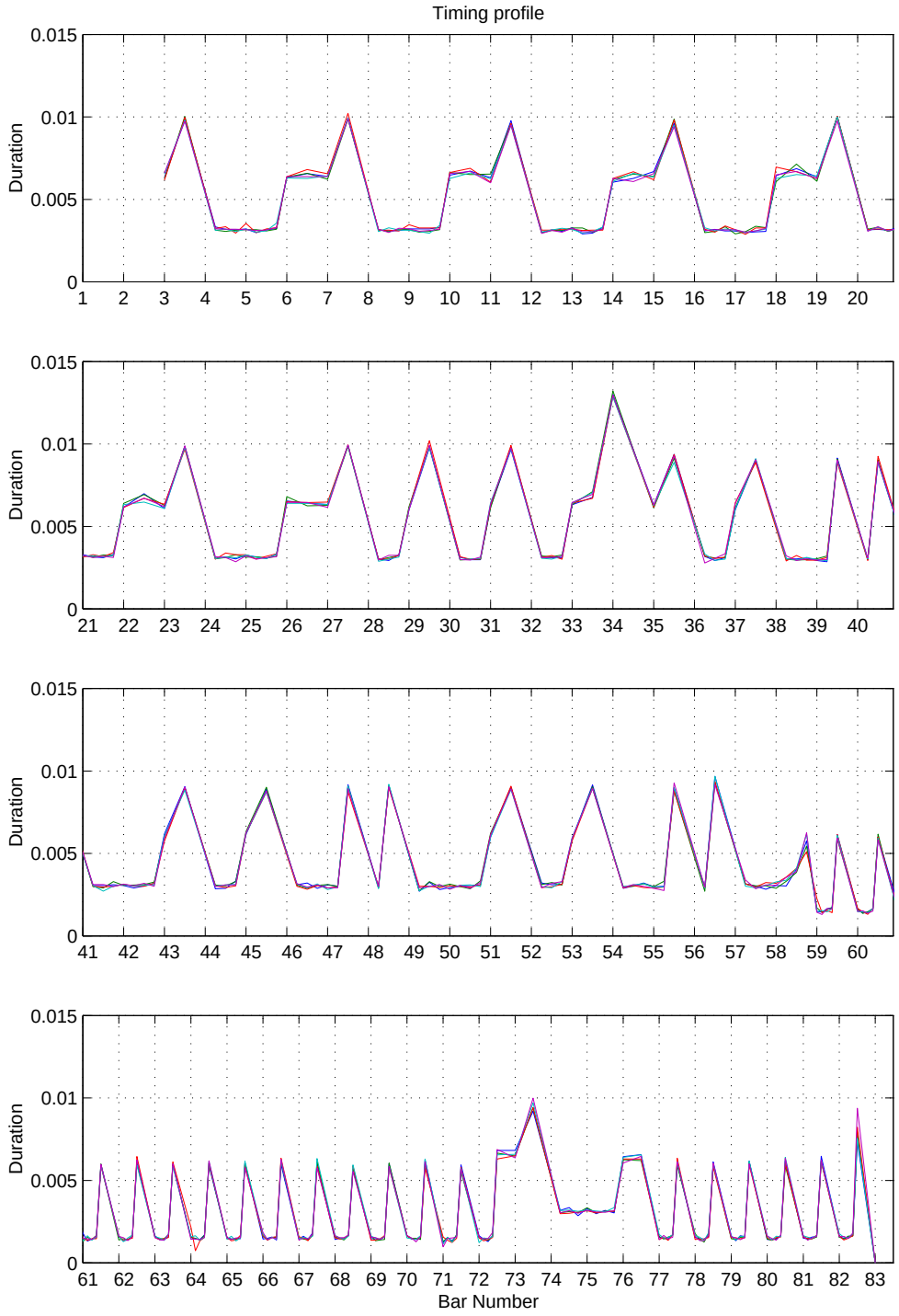


Figure 6.4: (colour) A graph showing the IOIs of the five performances of *Berceuse* after they have been scaled to the same total length.

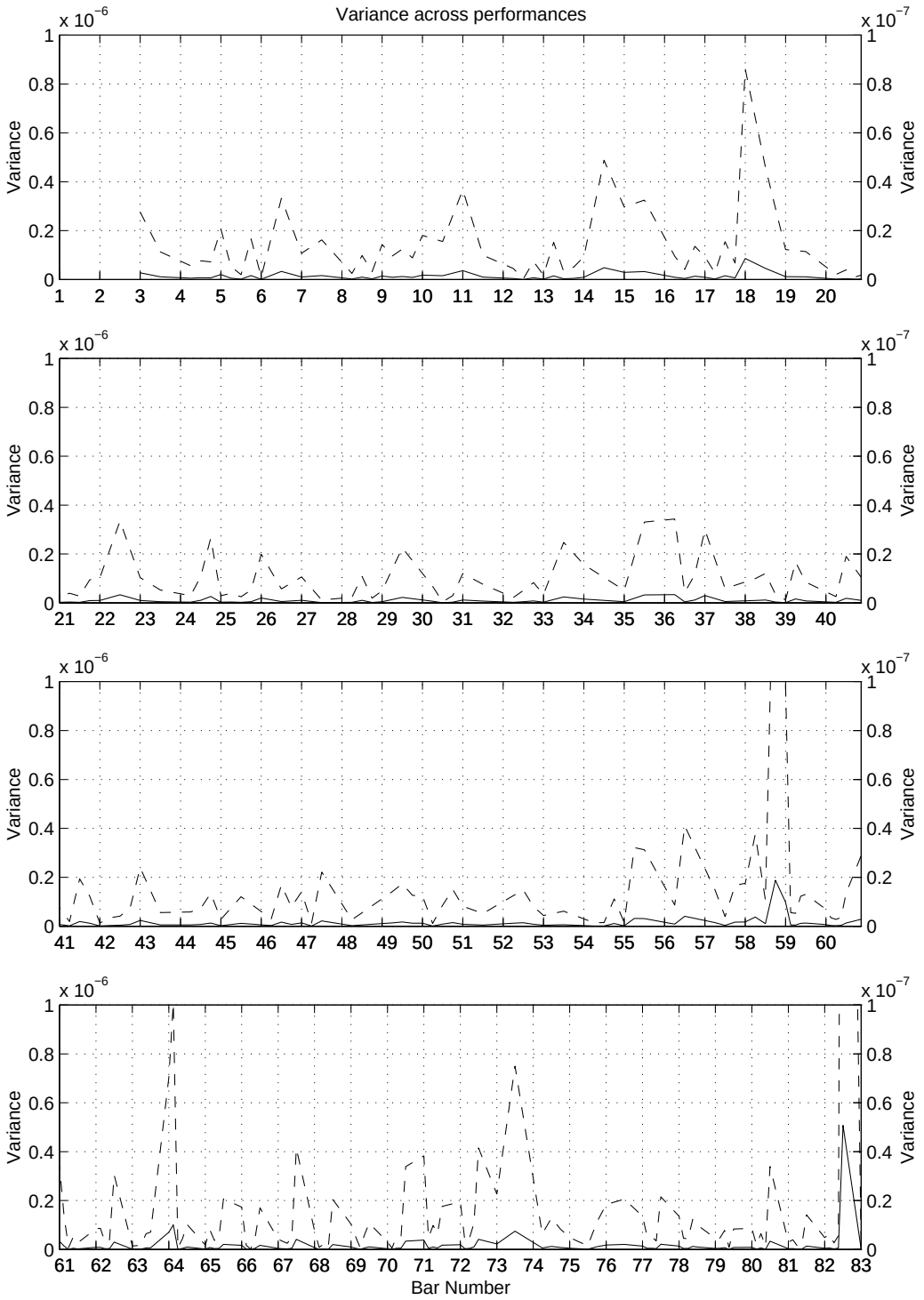


Figure 6.5: A graph showing the timing variance across the five *Berceuse* performances. The solid and dashed lines show the same variance information scaled by different amounts to show both the large and small-scale features. The scales for these two lines are presented on the vertical axes.

instructed to play *sempre dolce.*

The smaller peak at bar 43 corresponds to a hand written mark on the score, written by a previous performer, indicating the end of a swell. The peaks in bars 55, 56 and 58 correspond to the start of a swell, a *rallentando*, and then the end of the swell respectively. Also, at bar 59 the swapping of rôles occurs with the lead voice accompanying the second voice.

There are no clues in the score to explain the smaller peaks between bars 60 and 68. Again these could be due to performance errors. The peak in bar 70 coincides with a point in the score where the piece drops an octave.

The peaks in bars 73 and 76 match the points in the score where the first voice repeats the initial phrase for the last time, and also a *piano* annotation. Finally the last, and largest, peak occurs in bar 82 corresponding to the grand gesture of the performers to close the piece.

Table 6.2 shows the Pearson correlation coefficients (Hinton, 2001 p.256) of the durations for every pair of performances, including a performance made from averaging the IOIs. The Pearson correlation coefficient ($-1 \leq r \leq 1$) represents the correlation between two variables. A Pearson correlation of -1 indicates a perfect negative correlation, 0 represents no correlation and +1 represents a perfect positive correlation. In this case, the correlation coefficient is a measurement of how similar the IOIs of the same event in each of the pairs of performances are to each other. This is in turn a measurement of how similar the timing profiles are for each pair of performances.

The visual representation of these values can be seen in Figure 6.6 which shows, for every pair of performances, a scatter plot based on the durations of each event from each of the performances[6]. If two performances were identical, all the events would lie along a diagonal line with gradient 1 passing through the origin. The figure clearly shows a large similarity between all the performances which is confirmed by the correlations in Table 6.2.

The table shows that there are very strong correlations between all pairs of performances. The weakest correlation ($r = 0.9962$)[7] occurs between performances 3 and

[6]The four clusters of points which appear in the graphs correspond to the four most common note durations prescribed by the score.

[7]All the results for the Pearson correlation coefficient are significant ($p < 0.01$).

Performance	1	2	3	4	5	Avg. Perf.
1	1.0000	0.9981	0.9968	0.9981	0.9966	0.9991
2	-	1.0000	0.9971	0.9977	0.9968	0.9991
3	-	-	1.0000	0.9965	0.9962	0.9985
4	-	-	-	1.0000	0.9971	0.9990
5	-	-	-	-	1.0000	0.9971
Average	-	-	-	-	-	1.0000

Table 6.2: Correlation Coefficients (r) between the five recorded performances of *Berceuse* and the average performance.

5. Looking at the column of the table that represents the correlations between each performance and the average performance, the lowest correlation with the average performance ($r = 0.9971$) is still very strong.

6.5.5.4 Influence of Breaks on the Performances

The five performances which were chosen for this phase of the study originated from two of the three recording sessions. The first two performances came from the middle recording session and the remaining three came from the last session. If the pause between these sessions had an influence on the performances, the data should be able to be clustered into two groups of performances within which the pairs of performances have a greater similarity to each other than the similarity of the pairs which span the two groups.

In the case of the performances of *Berceuse*, the first group of performances would consist of performances 1 and 2 (the two performances from the middle session). The second group would consist of performances 3, 4 and 5 (the three performances from the final section).

To measure this similarity, a comparison between the average distances of the pairs of performances within groups and the average distances of pairs of performances between groups is made. If the pairs of performances within the groups are, on average, more similar than the pairs of performances between groups, then the grouping of the

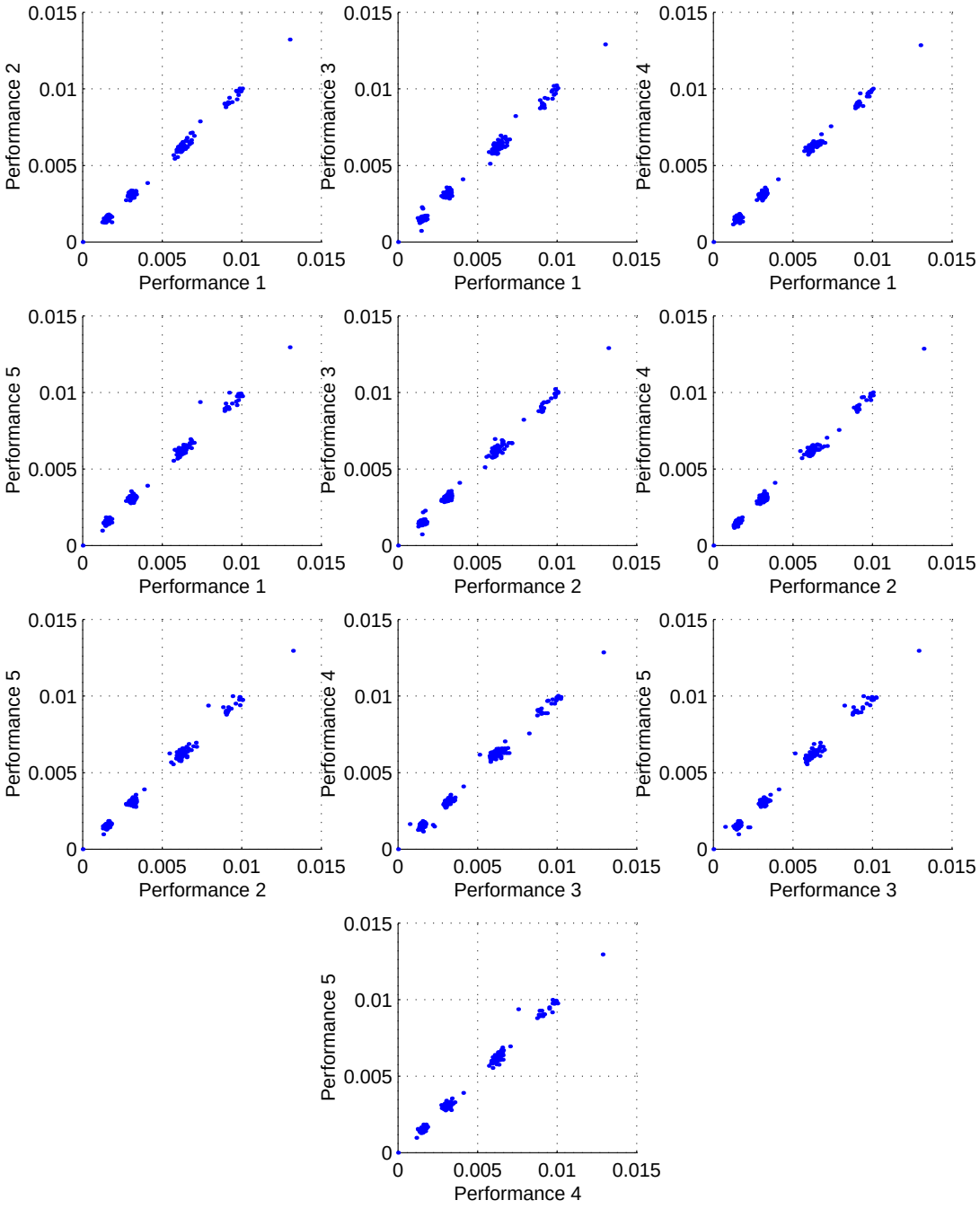


Figure 6.6: Scatter plots comparing the IOIs of the *Berceuse* performances.

performances is justified. If however, there is no significant difference between the distances of the performances within groups when compared with the distances of performances between groups then the grouping is artificial.

Table 6.3 shows the result of averaging the correlations of the performances[8] within the two groups and the result of averaging the correlations between groups. The within-group average correlation is $\mu = 0.9970$ and the between-group correlation is $\mu = 0.9972$. The similarity of these average correlations suggests that the grouping of the performances given above is artificial, which in turn implies that the pause between the sets of performances did not have a significant effect on the expressive timing of the performances.

Group	Pair	Correlation	Avg. Corr.
1	(1,2)	0.9981	
	(3,4)	0.9965	$\mu = 0.9970$
2	(3,5)	0.9962	$\sigma = 8.3815 \times 10^{-4}$
	(4,5)	0.9971	
	(1,3)	0.9968	
	(1,4)	0.9981	
	(1,5)	0.9966	$\mu = 0.9972$
n/a	(2,3)	0.9971	$\sigma = 5.9133 \times 10^{-4}$
	(2,4)	0.9977	
	(2,5)	0.9968	

Table 6.3: A table showing the within-group and between-group average correlations for the five performances of *Berceuse*.

6.5.6 Summary

The results of Phase I of this study show that there is a very strong correlation between performances of the same piece, which supports objective A. Now that the timing

[8] Where the correlation is used as the measure of similarity.

profiles have been shown to be consistent across these performances; the next phase of the study will try to achieve objective B.

6.6 Phase II

The goals of this phase of the study were to gather further evidence to corroborate objective A, but principally to gather more data (objective B).

6.6.1 Participants

For the second phase of the study, two professional keyboard players took part. They were both paid an appropriate fee for their participation.

Musician C a professional musician with over 10 years of experience.

Musician D a professional musician with over 8 years of experience.

6.6.2 Music

Two pieces from song cycles by Ludwig van Beethoven and Franz Schubert were used. Because the pieces were originally intended for a human voice accompanied by an instrument, the vocal part is monophonic and can therefore be analysed using GTTM.

The first piece was *Auf dem Hügel sitz ich spähend* from Ludwig van Beethoven's *An die ferne Geliebte* (Op. 98) composed between 1815 and 1816. *An die ferne Geliebte* (To the Distant Beloved) was composed around six poems by Alois Jeitteles (1764–1858) and is considered to be the first genuine song cycle ever to be composed. *Auf dem Hügel sitz ich spähend* (On the hill sit I, peering) is the first song from this collection and based upon a poem about two separated lovers.

The piece consists of five accompanied sections separated by two-bar interludes of accompaniment only. Each section lasts eight bars and corresponds to a stanza from the poem.[9] The sections share similar melodic and rhythmic structures.

[9]The word 'verse' is used in a poetic sense to denote a line of metrical writing. 'Stanza' is used to refer to a division of a poem into a series of metrical lines.

The second piece was Franz Schubert's *Gute Nacht* from the *Winterreise* (Op. 89) song cycle. *Winterreise* (The Winter's Journey) was composed in 1827 as two sets of twelve songs based on the poems of Wilhelm Müller (1794–1827). The poems describe the journey of a young man, who is disappointed in love, wandering through a bleak winter landscape. *Gute Nacht* is the first song of the set and the lyrics describe a man quietly leaving the home of his beloved because she has found someone new to love. As he leaves he writes "Good Night" in the snow at the gate to show he was thinking of her as he left.

The piece consists of three sections separated by large six-bar interludes. Each section, corresponding to each eight-verse stanza in the poem, has a six part structure.[10] Three motifs *a*, *b* and *c* are introduced and performed twice i.e. *aabbcc*. The two occurrences of *a* correspond to the first four verses of each stanza; the two repetitions of *b* correspond to the repeated fourth and fifth verses of each stanza and the two occurrences of motif *c* accompany the repeated last two verses of each stanza.

6.6.3 Equipment

The equipment was the same as described for phase I of the study (see Section 6.5.3).

6.6.4 Procedure

Musicians C and D spent only a short time rehearsing the recordings as they were confident in their own ability.

The musicians played each piece five times, swapping lead and accompaniment rôles between the two pieces (i.e. musician C played the lead for all five performances of the first piece and musician D played the lead for all performances of the other). Further to this swapping of rôles, the pieces were performed in a mixed order: the Schubert piece was played three times; the Beethoven three times; the Schubert another two times and then finally the Beethoven was played twice again. Between the first six performances and the last four, the musicians took a short break.

[10] Although there are four stanzas in the poem, because the music for the second stanza is an exact repetition of that for the first, only the music for three stanzas was performed.

The reasons for the breaks, and the mixed order of performances, was again to simulate a number of rehearsals made over a period of time (see Section 6.5.4).

6.6.5 Results

As before, the ten MIDI files (five for each piece) were inspected using Rosegarden. The professional ability of the two musicians was evident in the small number of errors that needed to be corrected. All of the performances were of a sufficiently high quality to be used for the consistency analysis.

6.6.5.1 Data Manipulation

The data was treated in the same way as described in Section 6.5.5.1.

6.6.5.2 Consistency Analysis of *Auf dem Hügel sitz ich spähend*

As can be seen in Figure 6.7 the graphs of the five performances of *Auf dem Hügel sitz ich spähend* are visually similar. However, it is immediately obvious that the performances are not as uniform as the performances of *Berceuse* presented earlier. The musicians were asked about the relative difficulty of the two pieces in this second phase of the study and they stated that this piece was the easier of the two. It may be that the relative simplicity of the piece gives the musicians more freedom to experiment with the timing during the performances.

Inspection of the performance durations, Table 6.4, shows that the maximum deviation in time from the average time is 3.65% and the maximum difference of performance duration is 8.16 seconds in a piece that is approximately 140 seconds in duration. Although this is significantly larger than the deviations encountered with the *Berceuse* performances, the performances are still very similar in duration and can be normalised together.

Figure 6.8 shows the timing variance of each event across each of the five performances. In contrast with the set of performances in Phase I, the variance for these pieces is distributed across a wide area of the graph rather than clustered at specific points. Examining the peaks in variance with the raw performance data shows that a

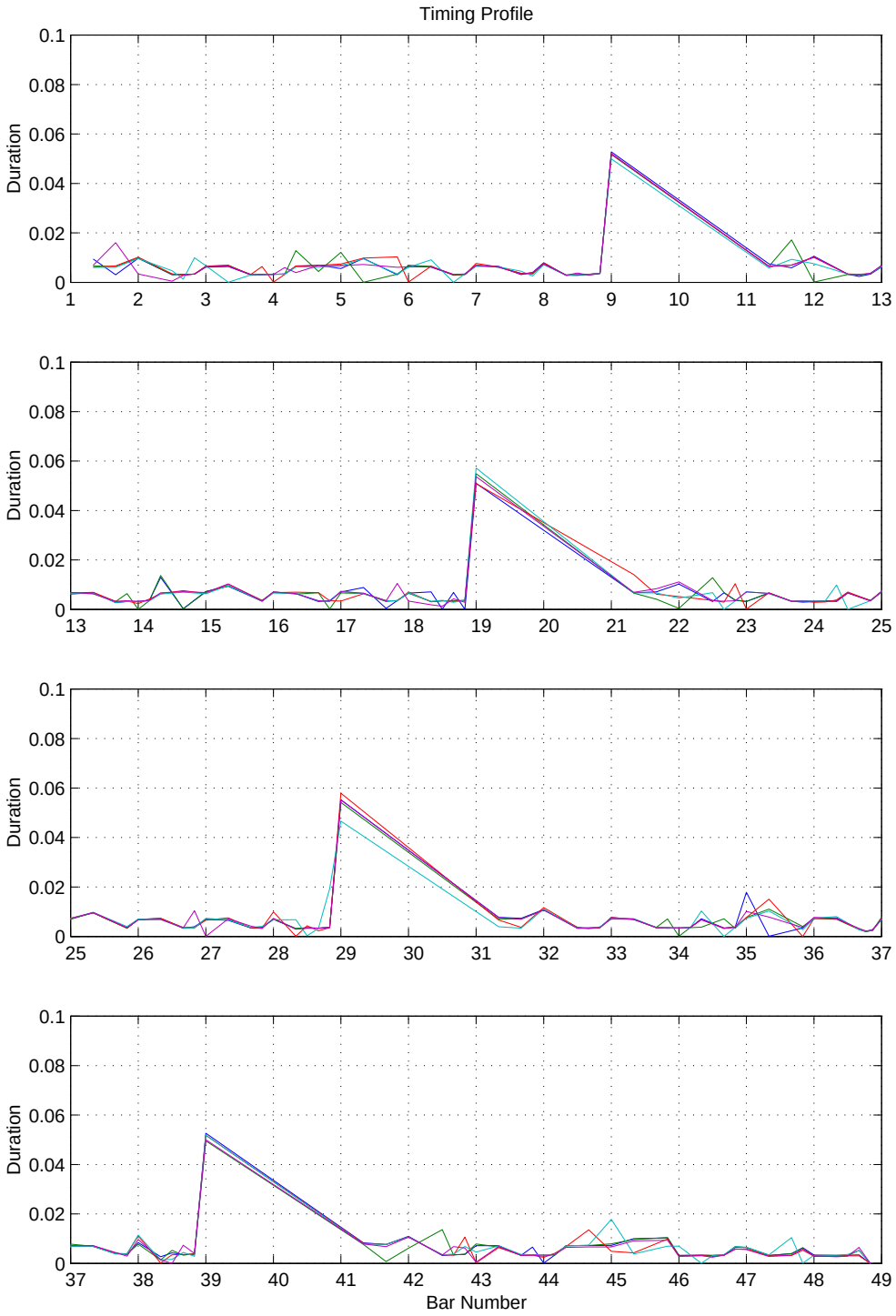


Figure 6.7: (colour) The five *Auf dem Hügel sitz ich spähend* performances

Performance	Start Time	End Time	Duration	% of Average	Difference
1	7.30	147.03	139.73	99.71%	-0.29%
2	4.53	144.35	139.82	99.77%	-0.23%
3	5.67	148.61	142.94	102.00%	+2.00%
4	5.97	140.99	135.02	96.35%	-3.65%
5	5.07	148.25	143.18	102.17%	+2.17%
	Arithmetic Mean $\mu = 140.14$				
	Standard Deviation $\sigma = 2.95$				

Table 6.4: Durations of the five Beethoven performances in seconds

large number of peaks are due to individual performances straying significantly from the timing suggested by the other performances at that point. For example, the peaks just before the start of bars 2 and 3 are primarily due to the deviations of one particular performance when the others at those points are very similar.

This pattern of individual performances accounting for the variance is by far the most common explanation for the individual peaks. However, it is not just one particular performance that is causing the majority of the deviations, each performance deviates at different points.

Calculating the Pearson correlation coefficients (Table 6.5 and graphically Figure 6.9) for these performances confirms that the overall similarity of the performances is less than the similarity obtained for the performances of *Berceuse*. Although the overall variance is large, the smallest (i.e. worst) pairwise correlation is $r = 0.9142$ which still shows a very high correlation.

When the average performance is compared with each of the individual performances, the weakest correlation is $r = 0.9296$ and the strongest is $r = 0.9787$ which strongly suggests the presence of an overall timing pattern which is followed by the musicians.

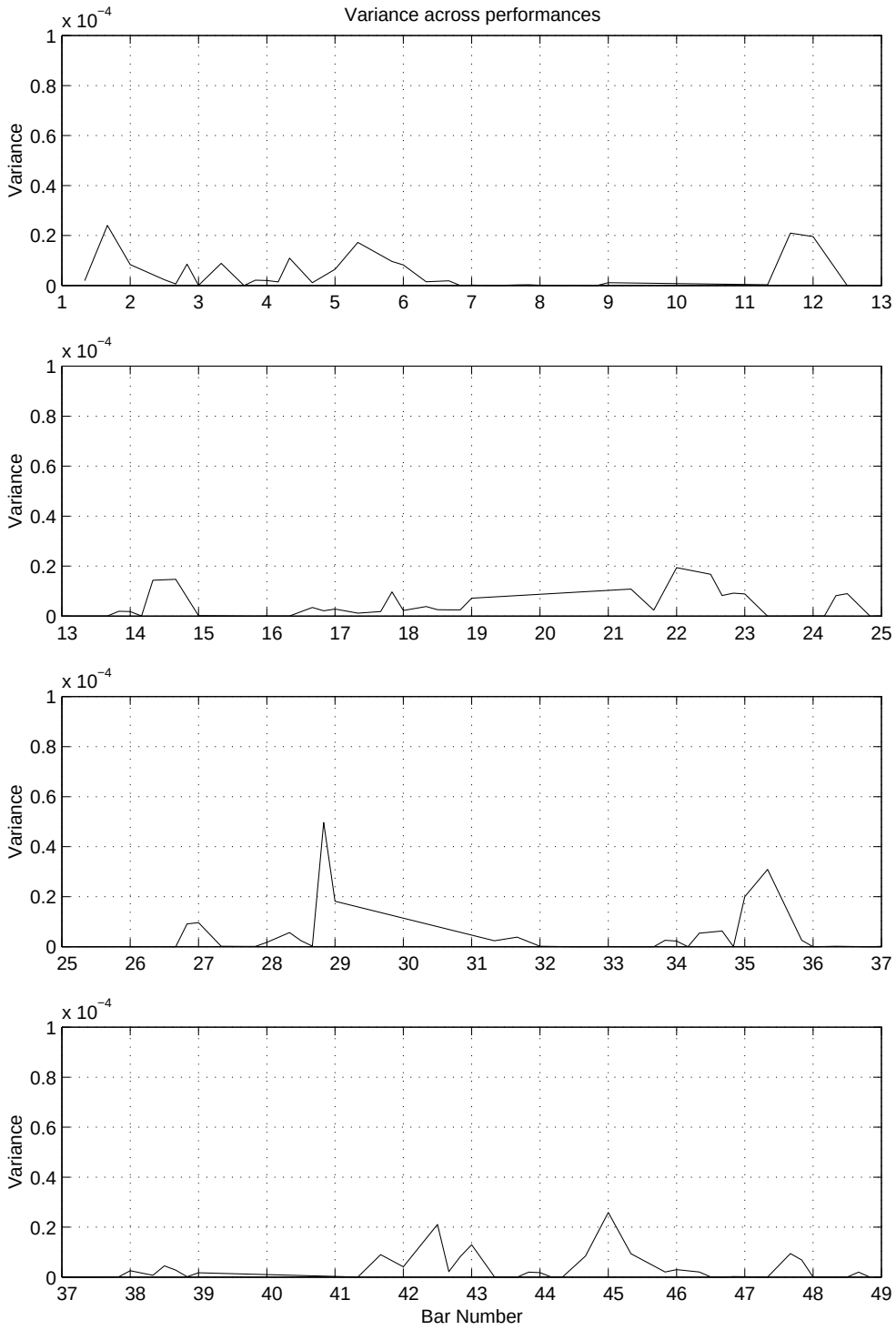


Figure 6.8: A graph showing the variance in timing across the five performances of *Auf dem Hügel sitz ich spähend*

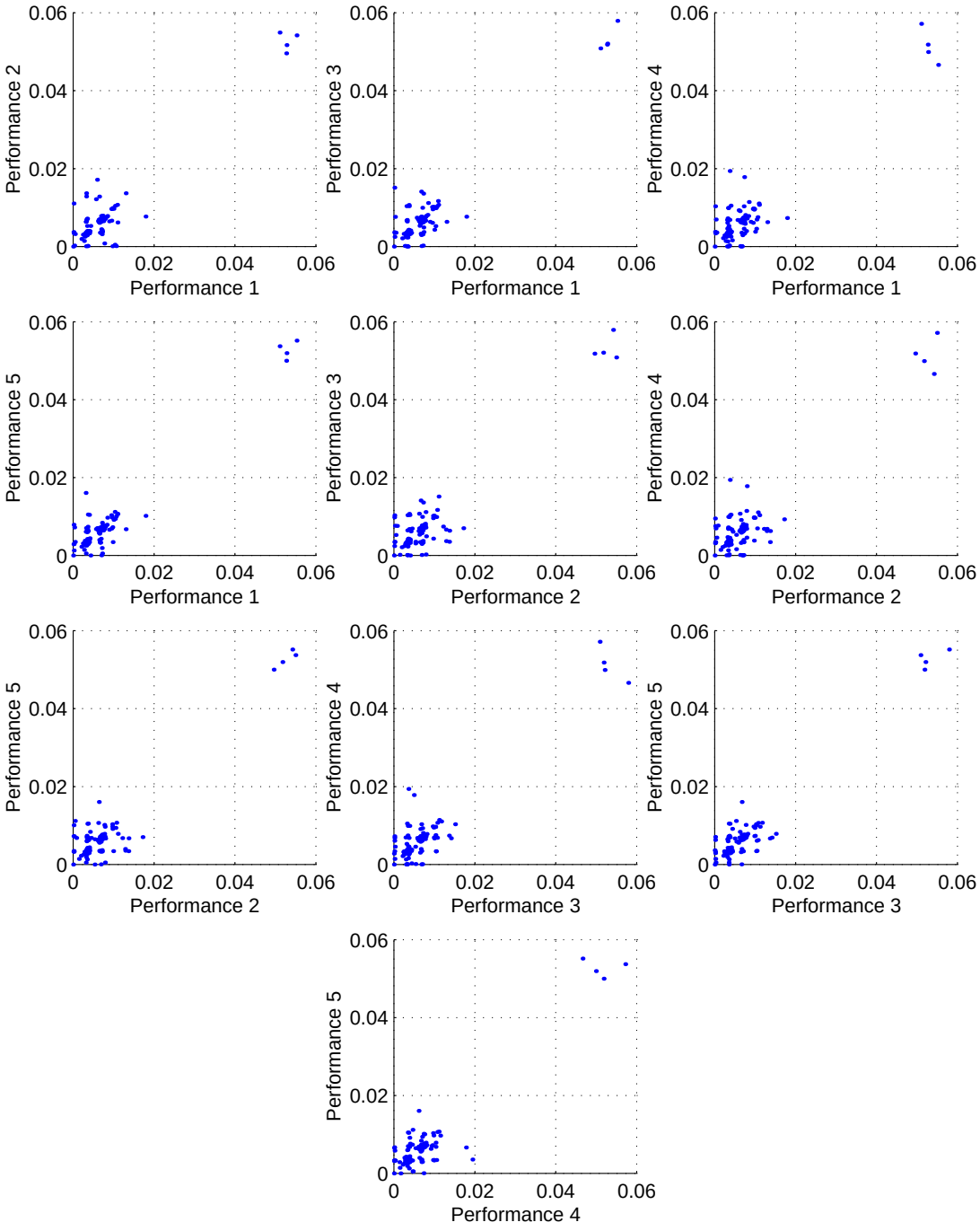


Figure 6.9: Scatter plots comparing the IOIs of the *Auf dem Hügel sitz ich spähend* performances.

Performance	1	2	3	4	5	Avg. Perf.
1	1.0000	0.9373	0.9463	0.9242	0.9573	0.9787
2	-	1.0000	0.9312	0.9142	0.9285	0.9677
3	-	-	1.0000	0.9278	0.9560	0.9781
4	-	-	-	1.0000	0.9296	0.9642
5	-	-	-	-	1.0000	0.9296
Average	-	-	-	-	-	1.0000

Table 6.5: Correlation Coefficients (r) between the five performances of *Auf dem Hügel sitz ich spähend* and the average performance.

6.6.5.3 Influence of Breaks on the Performances

As described earlier, one of the features of the experimental procedure was to insert pauses in the recording sessions to simulate the effect of pauses between rehearsal sessions. For the performances of *Auf dem Hügel sitz ich spähend*, the pause was introduced between the third and fourth performances. Using the hypothesis that the break did have an effect on the performances, then the similarity within the two groups of performances (i.e. performances 1, 2 and 3 as one group and performances 4 and 5 as another) should be greater than the similarity between the two groups (see Section 6.5.5.4 for more details).

Table 6.6 shows the average correlations for the within-group ($\mu = 0.9361$) and between-group pairings ($\mu = 0.9347$). Although the average distances are less similar than for the performances of *Berceuse*, the values are still very similar (especially when the range of correlations is taken into consideration).

6.6.5.4 Consistency Analysis of *Gute Nacht*

Figure 6.10 shows the five normalised performances of *Gute Nacht*. As with the performances of *Berceuse*, on initial visual inspection there is obviously a very strong similarity between the five performances.

The repeating structure of *aabbcc* can clearly be seen from the graph. Theme *a* occurs in bars 7–11 and bars 11–15. Theme *b* is performed in bars 15–19 and bars

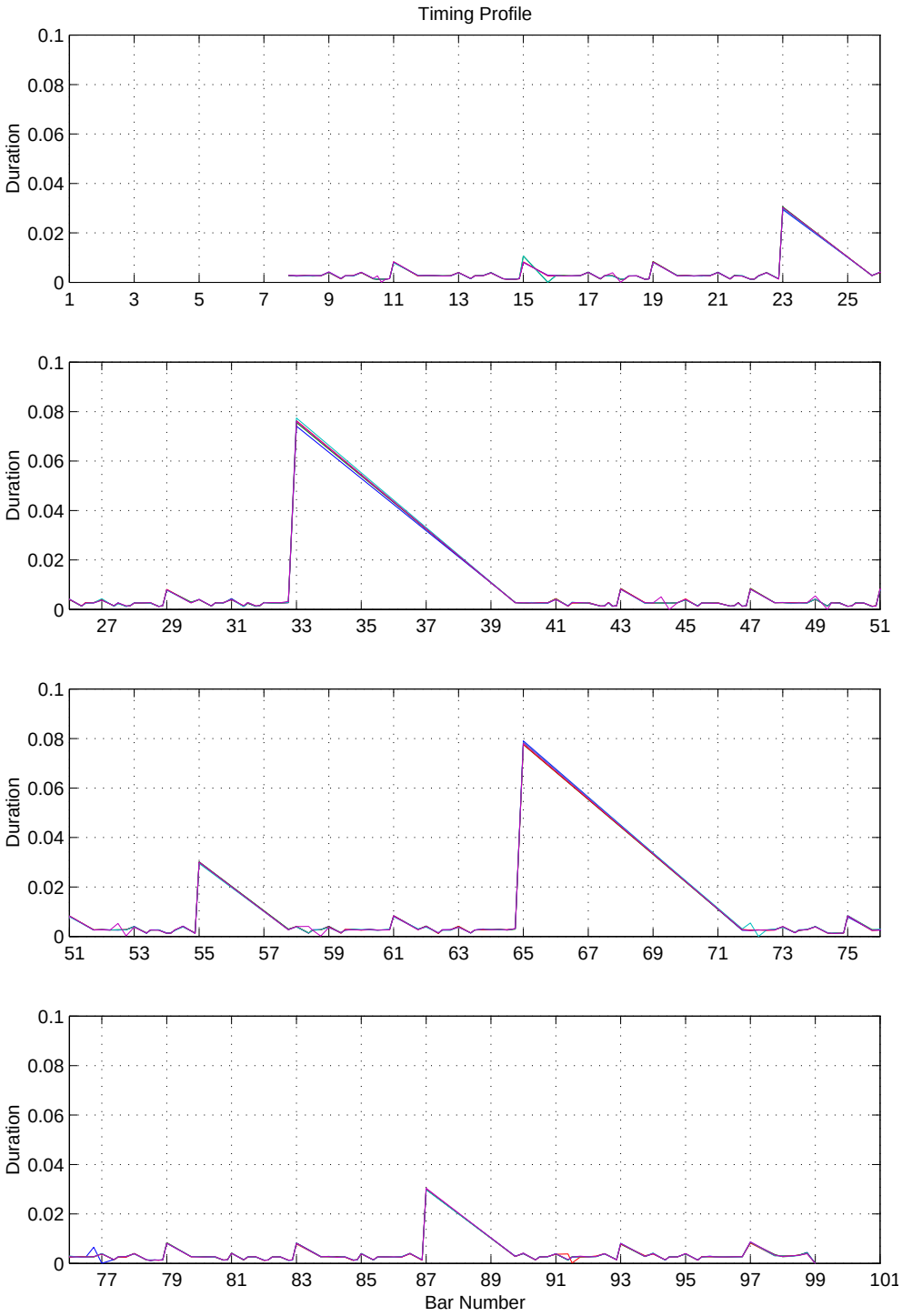


Figure 6.10: (colour) The five normalised performances of *Gute Nacht*

Group	Pair	Correlation	Avg. Corr.
1	(1,2)	0.9373	$\mu = 0.9361$ $\sigma = 0.0076$
	(2,3)	0.9463	
	(1,3)	0.9312	
2	(4,5)	0.9296	
n/a	(1,4)	0.9242	$\mu = 0.9347$ $\sigma = 0.0178$
	(1,5)	0.9573	
	(2,4)	0.9142	
	(2,5)	0.9285	
	(3,4)	0.9278	
	(3,5)	0.9560	

Table 6.6: A table showing the within-group and between-group average correlations of *Auf dem Hügel sitz ich spähend*.

19–23. There is then a relatively long pause of two bars followed by two repetitions of theme *c* in bars 25–29 and bars 29–33.

This structure described above is then repeated two more times starting in bar 39 and bar 71 separated by long six-bar pauses.

Table 6.7 shows that the overall durations of the pieces are very similar, with a maximum deviation of 1.63% from the mean and a maximum difference between any two performances of only 4.78 seconds.

Figure 6.11 shows the variance plotted at two different levels of detail to highlight both large scale deviations and lower-level deviations. A comparison of the performances plotted in Figure 6.10 with the graph of variances shows that the majority of the peaks can be accounted for by a single performance deviating from the others. For example, one performance is responsible for the peaks near bars 11, 18, 44, 49, 52 and 58.

Once the peaks created by individual deviations have been accounted for, the only remaining peaks are those near bars 15, 23, 33 and 65. The peak in variance near bar 15 occurs at the end of the first *aa* pattern and is due to a contrast between the

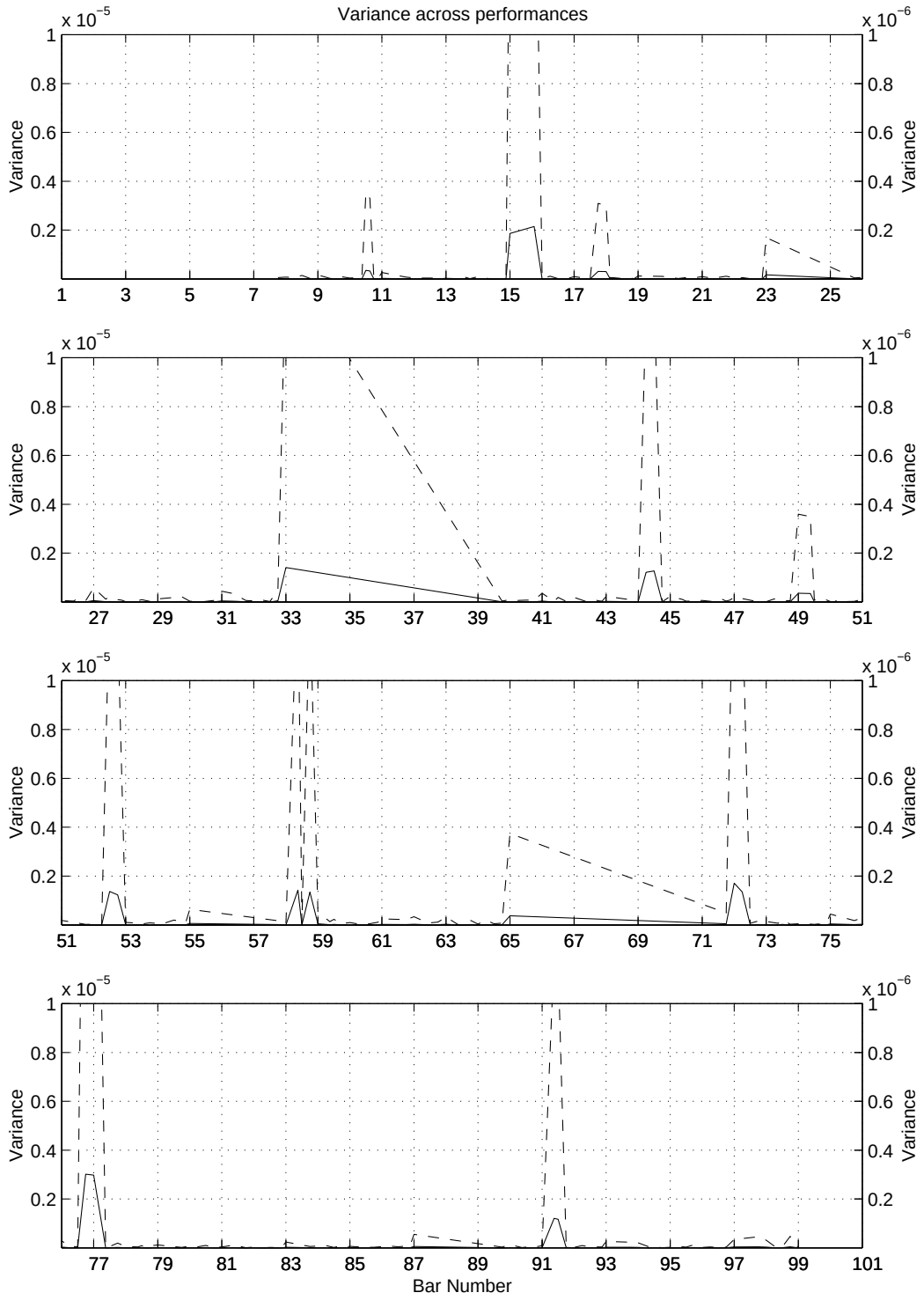


Figure 6.11: A graph showing the timing variance across the five performances of *Gute Nacht*. The solid and dashed lines show the same variance information scaled by different amounts to show both the large and small-scale features. The scales for these two lines are presented on the vertical axes.

Performance	Start Time	End Time	Duration	% of Average	Difference
1	19.68	252.88	233.20	99.27%	-0.73%
2	20.94	253.85	232.91	99.15%	-0.85%
3	23.25	257.21	233.96	99.60%	-0.40%
4	20.80	256.51	235.71	100.34%	+0.34%
5	27.11	265.85	238.74	101.63%	+1.63%
		Arithmetic Mean $\mu = 234.90$			
		Standard Deviation $\sigma = 2.15$			

Table 6.7: Durations of the five *Gute Nacht* performances in seconds

timing of two of the performances with respect to the other two. The peak near bar 23 corresponds to the two-bar pause at the end of the first *aabb* pattern of themes. The next two peaks near bars 33 and 65 correspond to the larger pauses between each of the three stanzas.

Table 6.8 shows the Pearson correlation coefficients of the five performances with each other and with an average performance (see Figure 6.12 for the graphical representation). As with *Berceuse*, there is a very strong correlation between all of the performances. The weakest correlation ($r = 0.9968$) occurs between performances one and five. The strongest correlation ($r = 0.9996$) occurs between the second and the average performances.

Performance	1	2	3	4	5	Avg. Perf.
1	1.0000	0.9981	0.9980	0.9973	0.9968	0.9989
2	-	1.0000	0.9989	0.9991	0.9976	0.9996
3	-	-	1.0000	0.9983	0.9976	0.9994
4	-	-	-	1.0000	0.9969	0.9992
5	-	-	-	-	1.0000	0.9969
Average	-	-	-	-	-	1.0000

Table 6.8: Correlation Coefficients (r) between the five performances of *Gute Nacht* and the average performance.

6.6.5.5 Influence of Breaks on the Performances

As with the other piece, the effect of the breaks between recording sessions on the performances of *Gute Nacht* will be examined. The recorded performances were again distributed around one break, with the first three performances taking place in the first recording session and the last two performances taking place in the final session.

Table 6.9 shows the results of calculating the average correlation both within-groups and between-groups. The within-group average correlation ($\mu = 0.9980$) and the between-group average correlation ($\mu = 0.9978$) are almost identical. This suggests that the break between the recording sessions did not have a significant impact on how the piece was performed.

Group	Pair	Correlation	Avg. Corr.
1	(1,2)	0.9981	$\mu = 0.9980$ $\sigma = 8.2209 \times 10^{-4}$
	(2,3)	0.9989	
	(1,3)	0.9980	
2	(4,5)	0.9969	
n/a	(1,4)	0.9973	$\mu = 0.9978$ $\sigma = 8.0850 \times 10^{-4}$
	(1,5)	0.9968	
	(2,4)	0.9991	
	(2,5)	0.9976	
	(3,4)	0.9983	
	(3,5)	0.9976	

Table 6.9: A table showing the within-group and between-group average correlations for the five performances of *Gute Nacht*.

6.6.6 Summary

Phase II of this study produced some interesting and surprising results. The performances of *Auf dem Hügel sitz ich spähend* contained more variance than was present in the performances of *Berceuse* and *Gute Nacht*.

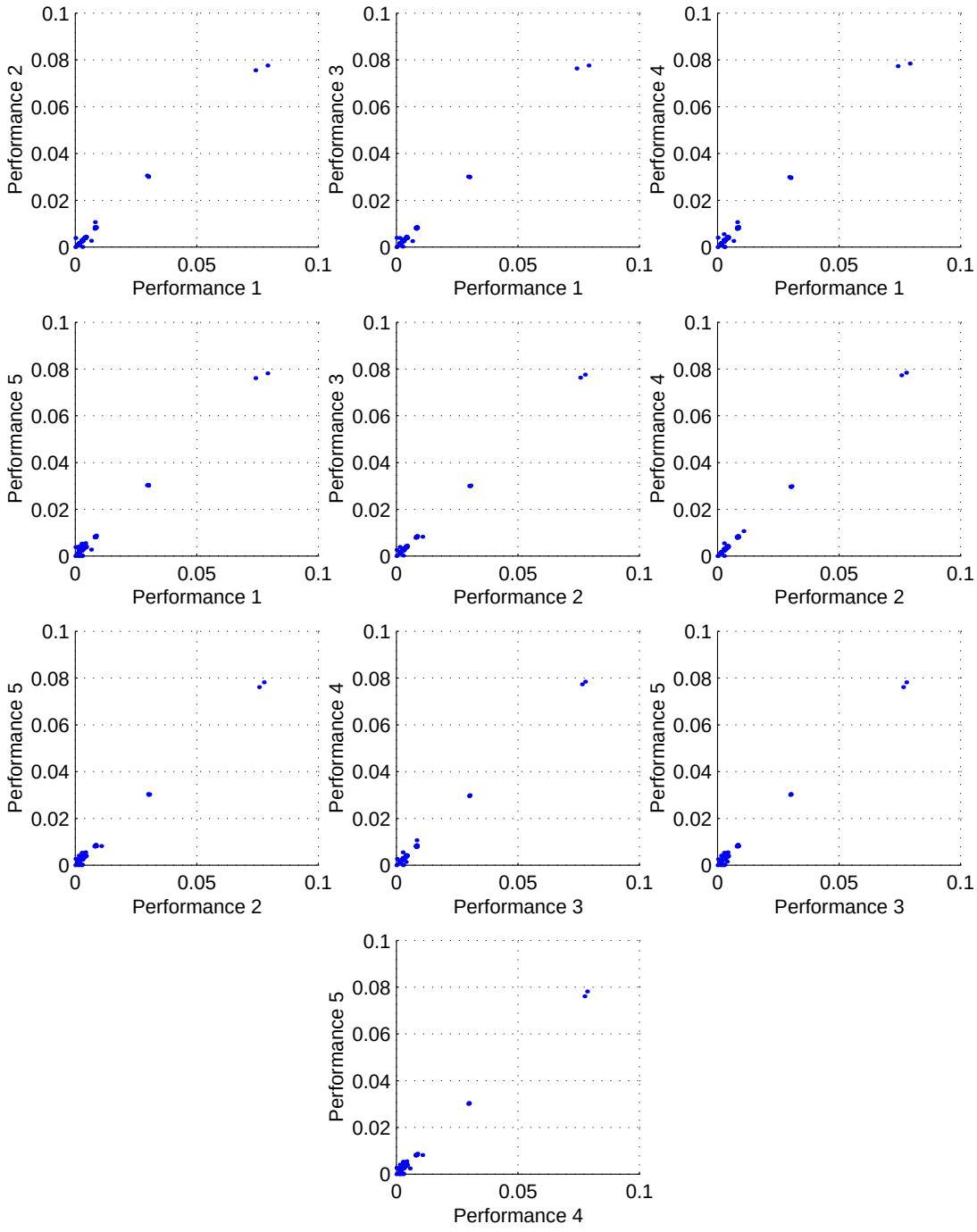


Figure 6.12: Scatter plots comparing the IOIs of the *Gute Nacht* performances.

A large number of the variations in the performances of *Auf dem Hügel sitz ich spähend* are not explainable from the score and the variances that occurred were of greater magnitude than in the performances of the other pieces. When this was discovered, musicians C and D were contacted and asked about the relative difficulty of the pieces. They stated that *Auf dem Hügel sitz ich spähend* was the easier of the two pieces. Perhaps the easier nature of the piece gave the performers more freedom to experiment with timing.

Despite the weaker correlation between the performances of *Auf dem Hügel sitz ich spähend*, the correlations are still relatively strong and provide further evidence to support objective A, that performances of the same piece by the same musician tend to be consistent.

However, the second objective (B) has clearly been satisfied and ten more performances have been gathered for subsequent analysis.

6.7 Summary

This chapter presented an empirical study of two phases to achieve two aims:

1. Corroborate Repp's (1996) conclusion that performers tend to be consistent, with regards to their expression, across performances.

2. Gather data for subsequent analysis.

The first phase of the study examined the expressive timing in five performances of *Berceuse* and found that across the five performances there was a strong similarity. A break which was introduced between some of the performances of the piece appeared to have little effect on the timing. These results provided supportive evidence of Repp's conclusion about consistency across performances.

The second phase of the study set out to further evidence to corroborate Repp's conclusion and to gather more performance data. Two different pieces were performed. The results of the timing analysis for *Auf dem Hügel sitz ich spähend* showed that the five performances were less similar to one another than was expected given the results from *Berceuse*. The nature of the piece increased the freedom of the performers to

experiment with the timing and this may have been the source of the greater differences in timing.

The second piece in the second phase of the study was *Gute Nacht*. The performances of this piece showed a similar degree of similarity to the performances of *Berceuse* with very high measures of correlation. The breaks between performances had no noticable effect on the timing.

The concrete output of this study is a set of fifteen MIDI files representing five performances of each of the three pieces. These recordings, since aim one was achieved, provide a set of data which can be used for further analysis. Therefore, aim two was also achieved.

The next chapter describes further analysis of the captured performances with the aim of providing information regarding the structure of the respective pieces.

Chapter 7

Performance Analysis

7.1 Introduction

Chapter 6 presented a study which was used to collect performance data from the performances of three different duets. The collected data contains the expressive timing introduced by the musicians. The aim of the research presented in this chapter is to apply analysis techniques to this collected data in order to find occurrences of interesting patterns in the performance which can be used to identify the structural properties of the piece.

Figure 7.1 illustrates the rôle of the research presented in this chapter within the structural disambiguation process. The process described in this chapter takes as input the rehearsal database of previous performances of the current piece and applies analysis techniques designed to detect certain features in these performances. The output from this component is a measure of where and how strongly these features were found in the performance data.

The resulting analysis of the performance data is then subjected to a further investigation in order to map the relatively disjoint features found by the performance analysis into a stronger set of features that can, in turn, be mapped onto the structural analysis (see Chapters 8 and 9).

The analysis presented in this chapter will identify structural features such as repetitions of timing patterns, phrase boundary points and, equally of interest, the points

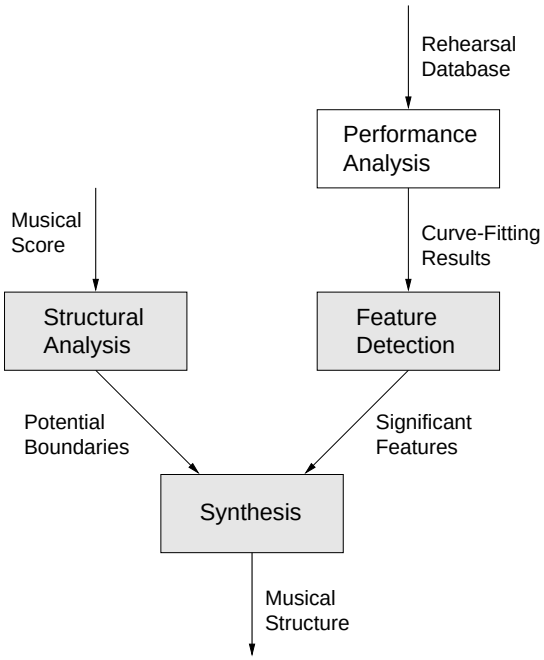


Figure 7.1: Pictorial representation of the rôle of this component within the structural disambiguation flow.

where no such features exist.

The following analysis techiques can be applied under the assumption that the music being analysed has a regular phrase structure and therefore a resulting regularity in the timing curve.

The two different analysis techniques used to process the performance data were:

- Autocorrelation – which correlates a time series with itself under a certain lag;
- Curve-fitting – which tries to find patterns of repeating curves in the data.

The autocorrelation analysis is particularly suited to finding occurrences of repetitive patterns within a time series. Autocorrelation does not return sufficient information to detect the locations of these patterns, only their size and relative strength. This technique is primarily used as a proven method to discover whether the repetitive structures which are to be identified are present in the data (Desain and Vos, 1990).

The novel curve-fitting analysis proposed in this thesis will identify instances of the specific convex curves that are indications of phrase-final lengthening and, unlike the autocorrelation analysis, positional information about where these occur in the data.

Both of these analysis techniques rely upon a discrete, but evenly spaced, stream of data; so before the analysis can proceed, the data from the musical performances needs to be interpolated.

7.2 Interpolation

The aim of interpolation is to complete the performance data so that there is an evenly distributed stream of data. For example, given the data:

$$D = (x_0, y_0), (x_1, y_1), (x_2, y_2), (x_4, y_4), (x_8, y_8), (x_{10}, y_{10})$$

The goal of the interpolation process is to complete the data set by finding y values for the data at x_3, x_5, x_6, x_7 and x_9 such that they respect the original data that was collected.

Knowing that the data represents a musical performance, there are a number of properties of the data which can be exploited in order to furnish the missing points. An example of some performance data can be seen in Figure 7.2, the data is presented

with the score time denoted on the x-axis and the performed event duration displayed on the y-axis.

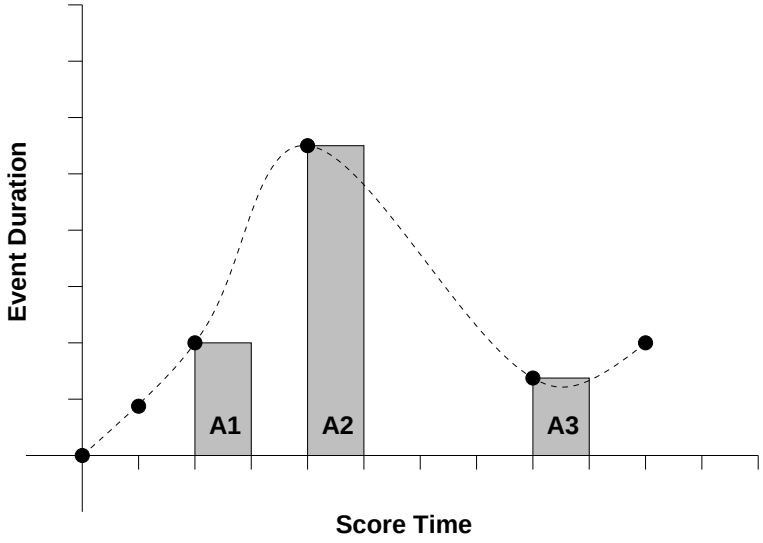


Figure 7.2: Graph showing event duration against score time for a performance.

The score time, x-axis, can be viewed as a non-contiguous source of data with each point on the axis corresponding to a possible musical event. A typical resolution for this axis would use a quaver or semi-quaver as the minimal unit.

The event duration, y-axis, is the actual amount of time a performed musical event lasted rather than the duration which was notated in the score. It is measured in terms of the amount of time between the start of the event and the start of the next event - the IOI.

Figure 7.2 shows an example performance of three events: A1, A2 and A3. Event A1 occurs after 2 units of score-time and has a performance duration of approximately 2 units. Event A2 occurs 2 units after A1 and has a performance duration of approximately 5.5 units. Finally, event A3 occurs 4 units of score-time after A2 and lasts approximately 1.5 units.

This data cannot be interpreted directly because not all measurements of the data correspond to events with the same weight. For example, if the data is interpreted directly, it would suggest the expressive timing curve shown as the dashed line in Figure 7.2. However, because the events were intended to have different durations,

as notated by the score, it is desirable to distribute the duration information across the graph. Without this distribution, it can be seen that the event A2 might have a disproportionately strong effect on the timing curve when compared with the other two events.

The following sections present two different interpolation approaches: the first is the most commonly used method (e.g. De Poli et al. (1998); Repp (1994a); Widmer (1997)); the second is the novel use of a standard interpolation technique.

7.2.1 Simple Interpolation

Figure 7.3 shows a graph with the duration information of the events distributed evenly across the score-time duration of the events. If the events had all been performed in a strictly mechanical manner with the events lasting exactly as prescribed in the score, the areas shown would all have the same height of one unit on the y-axis.

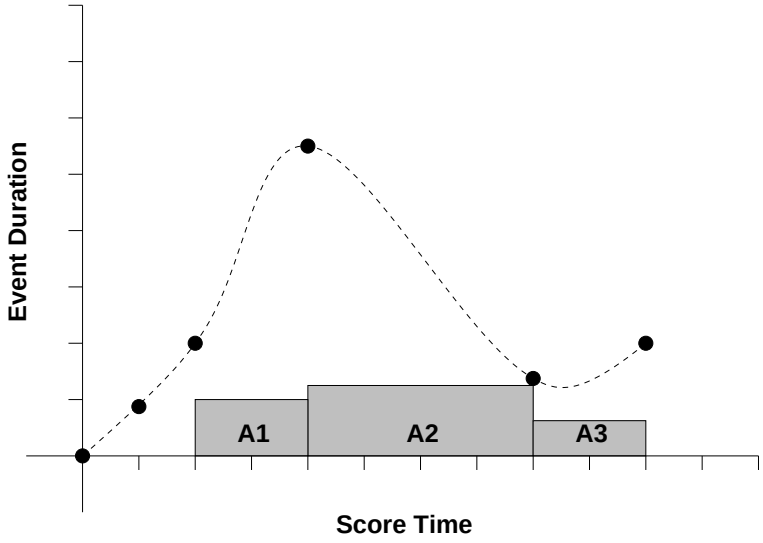


Figure 7.3: The results of distributing the event durations over score time using simple interpolation.

However, as can be seen from the figure, the durations when distributed across the score-time show a marked difference across the three events. Event A2 has lasted significantly longer than was required in the score, and event A3 has lasted significantly

shorter than prescribed.[1]

The results shown in Figure 7.3 give an approximation of the overall change in durations for each event. The resolution of this approximation is based upon the number of events and their frequency. This can lead to step-wise changes which makes the shape of the performance envelope blocky. For example, the graph clearly shows that the area occupied by A2 in the graph is disproportionately larger than the areas occupied by A1 and A3. The resulting envelope shows stepwise jumps and it is unlikely that the expressive performance would be based on such coarse steps. Rather, the performance will probably be based upon smoothly varying curves that would extend throughout the duration of an event. This leads to the novel application of the interpolation method described below.

7.2.2 Context-based Interpolation

In contrast to the above technique, the interpolation process proposed in this research takes into account the surrounding context in order to provide the missing values.

The first step of this process is to replace the rectangular areas of the events which would have been produced by the method described above with a series of points (see Figure 7.4). These points are placed at the required resolution and will be used to provide a finer level of detail.

The points $(x_2,y_2)\ldots(x_9,y_9)$ need to be chosen in such a way as they preserve the information that is known and provide a smoother and more uniformly distributed representation of the timing curve.

If the timing curve for the whole performance is modelled as a series of smaller, connected curves; it is possible to apply a set of constraints to the points $(x_2,y_2)\ldots(x_9,y_9)$ in such a way as to fulfill the desired properties of the points and to respect the information that is already known.

At this point, some terminology will be introduced to describe the classes of data point that occur:

Fixed-point is an (x,y) pair which is known to be accurate at the current resolution

[1] This is apparent because the height of the rectangle representing event A2 is above the line of $y = 1$. Similarly for the event A3.

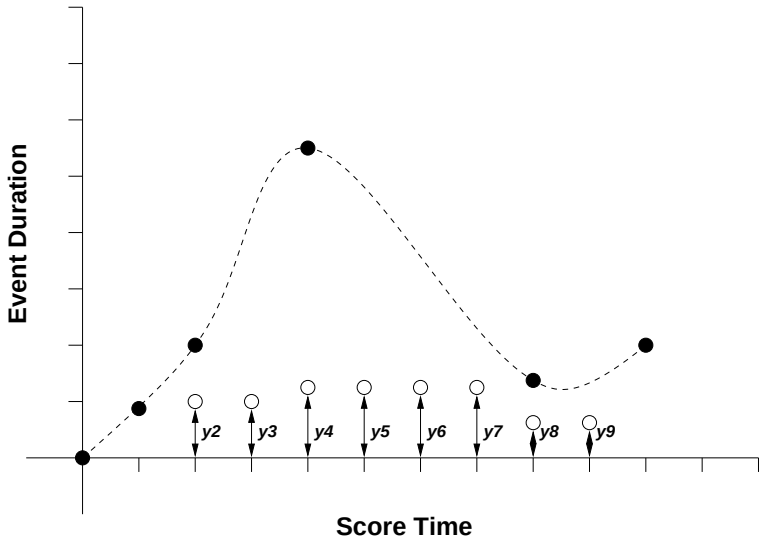


Figure 7.4: Replacing the actual durations with a uniformly distributed set of points.

and should not be changed in any subsequent calculations. For example, if the piece consisted of a continuous stream of quavers and the x-axis resolution was the quaver unit, all the points would be fixed-points.

Pivot-point is an (x,y) pair which occurs at the cusp between two possible curves, it is called a pivot point because as the value of the y coordinate of this point changes it affects both the curve ending at this point and the one beginning from it.

Variable-point is an (x,y) pair about which the least amount of information is known and is the main constituent of the missing data between fixed and pivot points.

In Figure 7.4: point (x_1,y_1) is a fixed-point, the points $(x_2,y_2),(x_4,y_4),(x_8,y_8)$ are pivot-points and all the other points are variable-points. The variable-points and pivot-points can potentially be adjusted (vertically) by the interpolation process.

A curve, which can be represented with quadratic equation, is placed between pairs of pivot points. A quadratic equation is used to represent the timing curve during the interpolated segment of the piece because it is the simplest polynomial that allows a single change in direction of timing durations to occur.[2]

[2]The quadratic curve allows the interpolation to model the performer accelerating, decelerating,

Previous research has used parabolic curves to model such timing curves (Todd, 1989c). In this research, quadratic curves are used to closely approximate the results which would be available from a parabolic curve-fitting model.

The equation representing the curves will have the form shown in Equation 7.1.

$$f(x) = ax^2 + bx + c \tag{7.1}$$

Given the three pivot points in Figure 7.4, it is possible to state separate equations for each curve, see Equations 7.2 to 7.4.

$$f_a(x) = a_a x^2 + b_a x + c_a \quad \text{for} \quad x_2 \ldots x_4 \tag{7.2}$$

$$f_b(x) = a_b x^2 + b_b x + c_b \quad \text{for} \quad x_4 \ldots x_8 \tag{7.3}$$

$$f_c(x) = a_c x^2 + b_c x + c_c \quad \text{for} \quad x_8 \ldots x_9 \tag{7.4}$$

Referring to Figures 7.2 and 7.4 it can be seen that the sum of *y* values for the data points between two pivot points must sum to the original *y* value recorded for that pivot point. This information leads to Equations 7.5 to 7.7.

$$\mathsf{A1} = \sum_{x=2}^{3} f_a(x) \tag{7.5}$$

$$\mathsf{A2} = \sum_{x=4}^{7} f_b(x) \tag{7.6}$$

$$\mathsf{A3} = \sum_{x=8}^{9} f_c(x) \tag{7.7}$$

At this point, there is not sufficient information to find solutions to Equations 7.2 to 7.7. Further to this, the current set of equations describe the points representing each part of the curve as individual entities. Further restrictions to the possible values the points may take can be added by considering how the curves interact.

To provide a smooth curve over the entire piece, it is necessary to join the distinct curves together as smoothly as possible. To do this, two more sets of continuity constraints are applied.

First, to avoid disjoint curves, another restriction is added that sets the intersection of two curves to occur at the pivot points. In other words, the equations of the two

accelerating then decelerating or decelerating then accelerating.

curves either side of a pivot point must equal the same value at that pivot point. Equation 7.8 links the curve spanning $x = 2,3$ to the curve spanning $x = 4\ldots7$ at $x = 4$. Similarly, Equation 7.9 links the curve that spans $x = 4\ldots7$ to the curve spanning $x = 8,9$ at $x = 8$.

$$f_a(x_4) = f_b(x_4) \tag{7.8}$$

$$f_b(x_8) = f_c(x_8) \tag{7.9}$$

Finally, to ensure the transition between curves is as smooth as possible, an extra set of conditions is added to ensure that the derivatives (i.e. the gradients) of the two curves at the pivot points are equal (see Equations 7.10 and 7.11).

$$\begin{aligned} \frac{df_a(x_4)}{dx} &= \frac{df_b(x_4)}{dx} \\ 2a_a(x_4) + b_a &= 2a_b(x_4) + b_b \end{aligned} \tag{7.10}$$

$$\begin{aligned} \frac{df_b(x_8)}{dx} &= \frac{df_c(x_8)}{dx} \\ 2a_b(x_8) + b_b &= 2a_c(x_8) + b_c \end{aligned} \tag{7.11}$$

The constraints described above lead to a solution similar to that shown in Figure 7.5. As can be seen from the figure, the resulting curve is smooth and exploits all the information which could be gathered from the initial measurements.

The quadratic interpolation method as presented above was chosen as it is the simplest approach which is flexible enough to model a single change in acceleration direction and which makes full use of the available information. Further work could investigate whether it would be possible for splines to be used to provide the interpolated data. Importantly, if splines were used, a similar model of distributing the weight of an event over its duration would need to be a factor in the shape of the resulting curves.

The following sections describe the results of applying both of these interpolation processes to the recorded performances of the three pieces.

7.2.3 Interpolation: *Berceuse*

Figure 7.6 shows the result of applying both the simple interpolation method and the context-based interpolation to the performances of *Berceuse*. The graph shows the

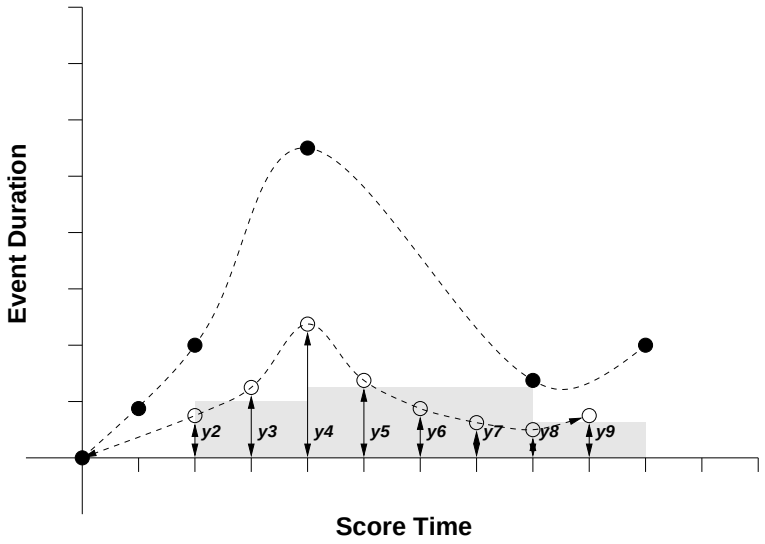


Figure 7.5: Smoothing the uniformly distributed points (the results of the simple interpolation are shown in grey for comparison).

average of the five simple interpolations as a dashed-line and the average of the context-based interpolations as a solid line.

As can be seen from the figure, the majority of peaks have been smoothed when compared with the original performances (see Figure 6.4) by both of the interpolation processes. The results from the two processes seem generally very similar, with some deviations due to the context-based interpolation offering a overall smoother curve.

The first point of deviation between the two curves is at the beginning of the piece, where the context-based interpolation begins with a large oscillation from bars 3–5. This large oscillation is due to the initial starting point of the interpolated curve starting at zero which is then gradually damped during the first few bars of the piece. The curves are then very similar up to bar 73 with few exceptions.

In the first 73 bars there are some good examples of the smoothness of the context-based method. At bars 20, 23, 34 and 57 there are examples of the sharp step-wise motion of the simple interpolation method which are smoothed by the context-based method.

There is another large variation between the two curves from bars 72 to 75. This corresponds to the point in the last third of the performance when the first voice tem-

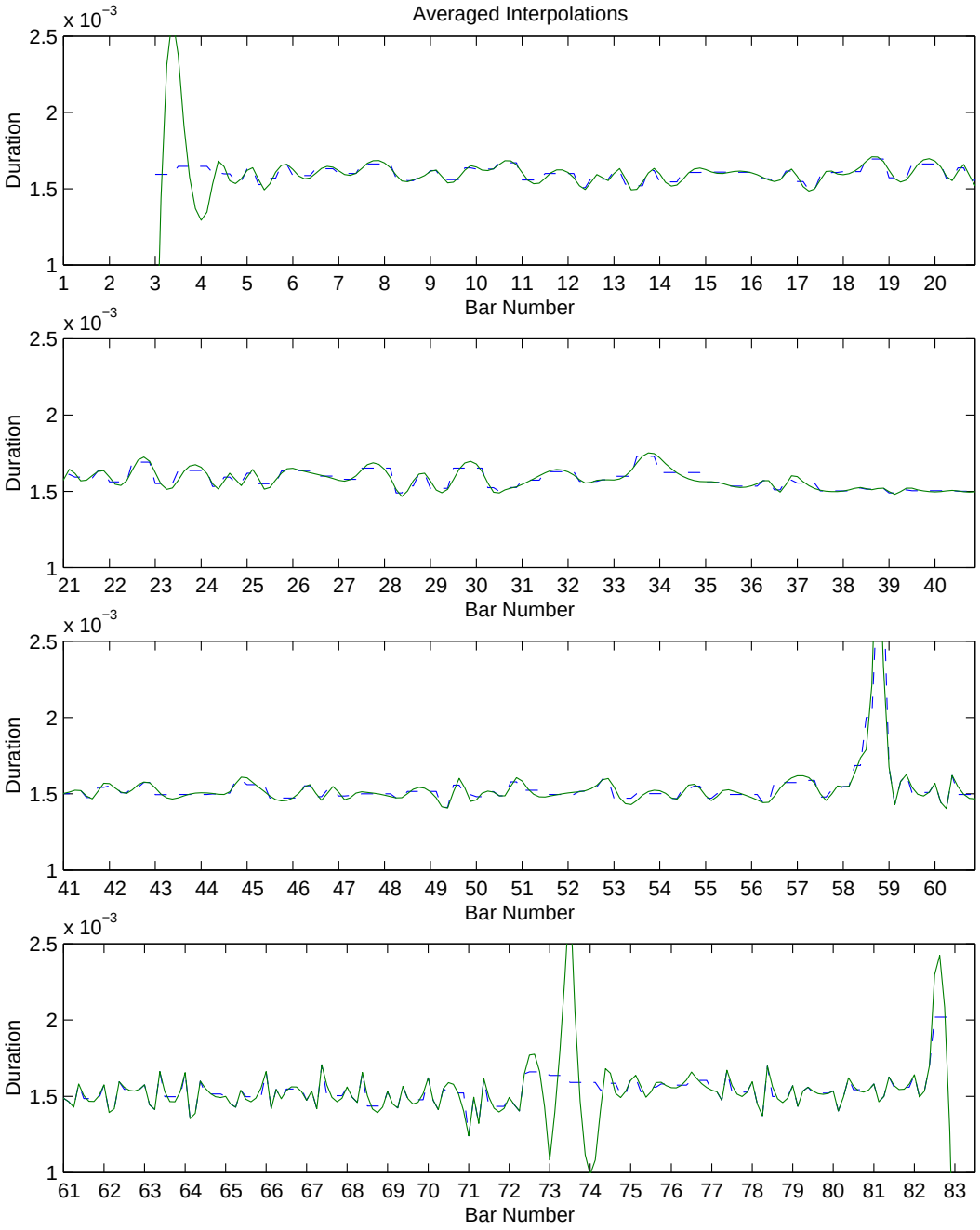


Figure 7.6: A graph showing the results of the simple interpolation (dashed-line) and context-based interpolated (solid-line) performance IOIs of *Berceuse*.

porarily regains the melodic phrase presented in the first third of the piece. The simple method of interpolation presents a mostly flat set of timings in this area whereas the context-based interpolation offers another example of an oscillating tempo curve.

The peak in bar 58 occurs at the end of the transitionary middle phase of the piece, just before the lead voice adopts the rôle of the accompanist, after a four-bar long swell. The peak in bar 82 cannot be explained from the score but is probably due to a final gesture at the end of the piece.

Overall, the resulting curves from the two methods are very similar, $r = 0.6494$ $(N = 640,\ p < 0.01)$, but the context-based method offers more instances of the desired smooth transitions in timing. The context-based method does, however, suffer when the context suggests an extreme positive or negative change in timing which, because subsequent points depend on the previous results, has to be compensated for during the next few bars of the piece and results in the previously mentioned oscillations.

7.2.4 Interpolation: *Auf dem Hügel sitz ich spähend*

Another weakness of the context-based interpolation became apparent whilst applying it to the performance IOIs of *Auf dem Hügel sitz ich spähend*. Unlike *Berceuse*, *Auf dem Hügel sitz ich spähend* has a number of long pauses in the piece (e.g. starting at the second beat of bar 9 through to the second beat of bar 11). During these pauses, the context-based interpolation had little context to work with and produced results which were clearly incorrect and which led to artifacts during the subsequent analysis process.

To compensate for these occurrences, whenever the context-based interpolation process encounters a rest that lasts two bars or longer, the simple interpolation process is used to interpolate that section of the piece. The context-based interpolation then continues from the end of the rest until it encounters another long rest, at which point it uses the simple interpolation process again.[3]

Applying the two interpolation processes to the performances of *Auf dem Hügel sitz ich spähend* results in the performance curves presented in Figure 7.7.

[3] Although the context-based interpolation process now includes some results from the simple interpolation, the process will still be referred to as context-based interpolation.

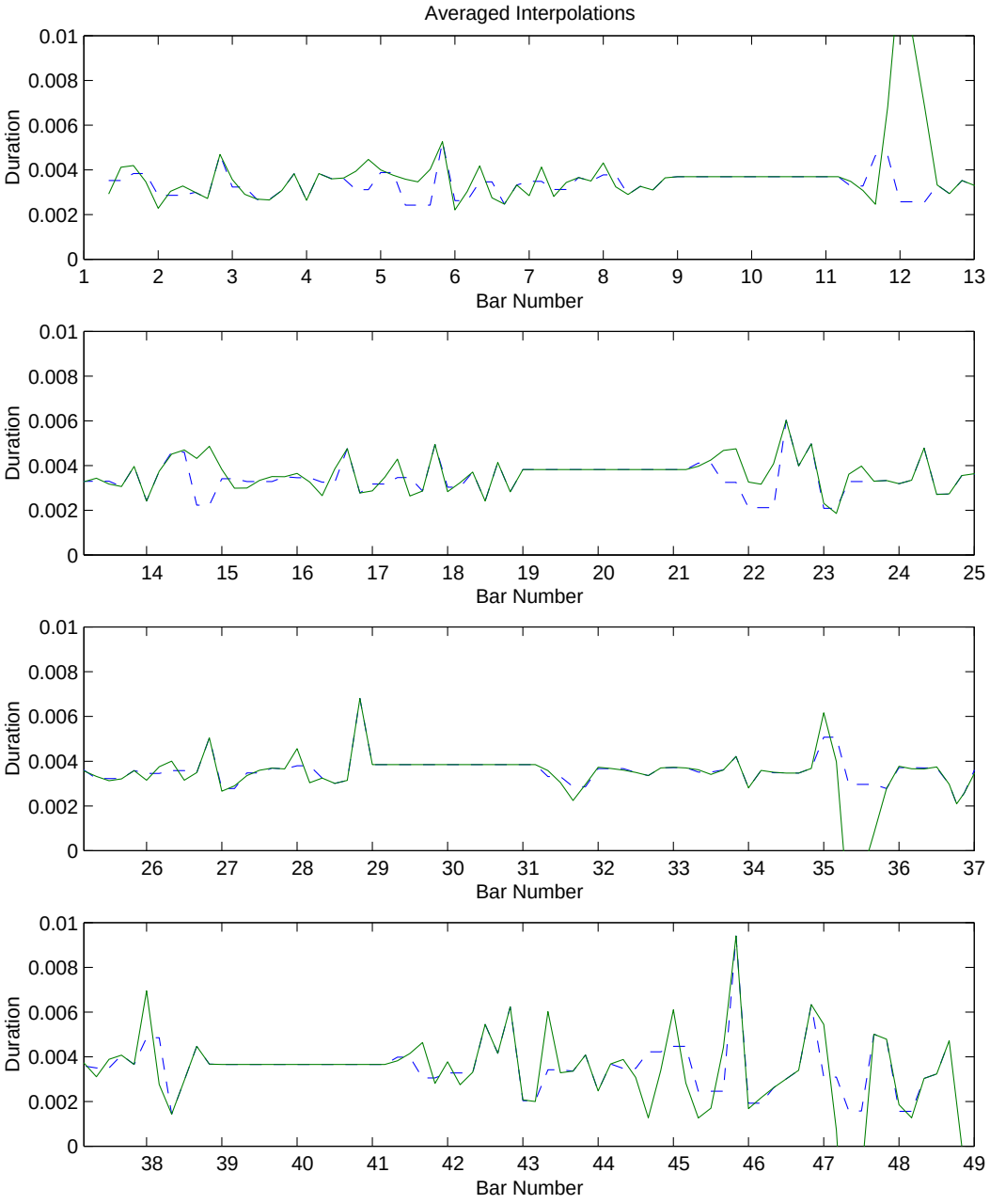


Figure 7.7: A graph showing the results of the simple interpolation (dashed-line) and context-based interpolated (solid-line) performance IOIs of *Auf dem Hügel sitz ich spähend*.

The resulting curves are less similar than for the interpolations of *Berceuse* ($r = 0.5343,\ N = 286,\ p < 0.01$).

There are some parts of the context-based interpolation which are not musically sensible, for example at bar 35 and bar 47 the interpolated curves dip below the x-axis making the durations of musical events at those points negative.

Other points of interest occur at bars 12 and 14 where the context-based interpolation suggests a timing change that opposes the results of the simple interpolation method. Bar 5 shows an example of the desired smoothing ; the simple interpolation suggests a number of sharp steps which are translated into a smoother transition by the context-based interpolation.

As with the interpolation of *Berceuse*, the results of the context-based interpolation are varied. The overall shape of the context-based curve is smoother and offers some sensible interpolations but there are some cases where the method produces surprising and non-musical results.

7.2.5 Interpolation: *Gute Nacht*

Figure 7.8 shows the result of applying the interpolation processes to the performances of *Gute Nacht*. The resulting interpolations are again very similar ($r = 0.5795,\ N = 730,\ p < 0.01$).

Examination of the graphs show that there are a number of points where the simple and context-based method deviate. The majority of these points occur at positions in the score where there are rests. These rests occur at bars 11, 15, 19, 23 and 29 for the first stanza and at similar intervals for the second and third stanzas of the song.

Apart from these deviations, there are two other points where there is a significant difference between the two interpolation methods. The first one occurs at the start of the piece, again the application of the context-based interpolation produces an oscillating time curve that takes a bar to settle. The second occurs in bar 77 where the truncated peak and trough of the simple interpolation is replaced with a taller, wider peak by the context-based interpolation.

As before, apart from the cases mentioned above, the two interpolations offer similar results but with the context-based interpolation giving a smoother form to the timing

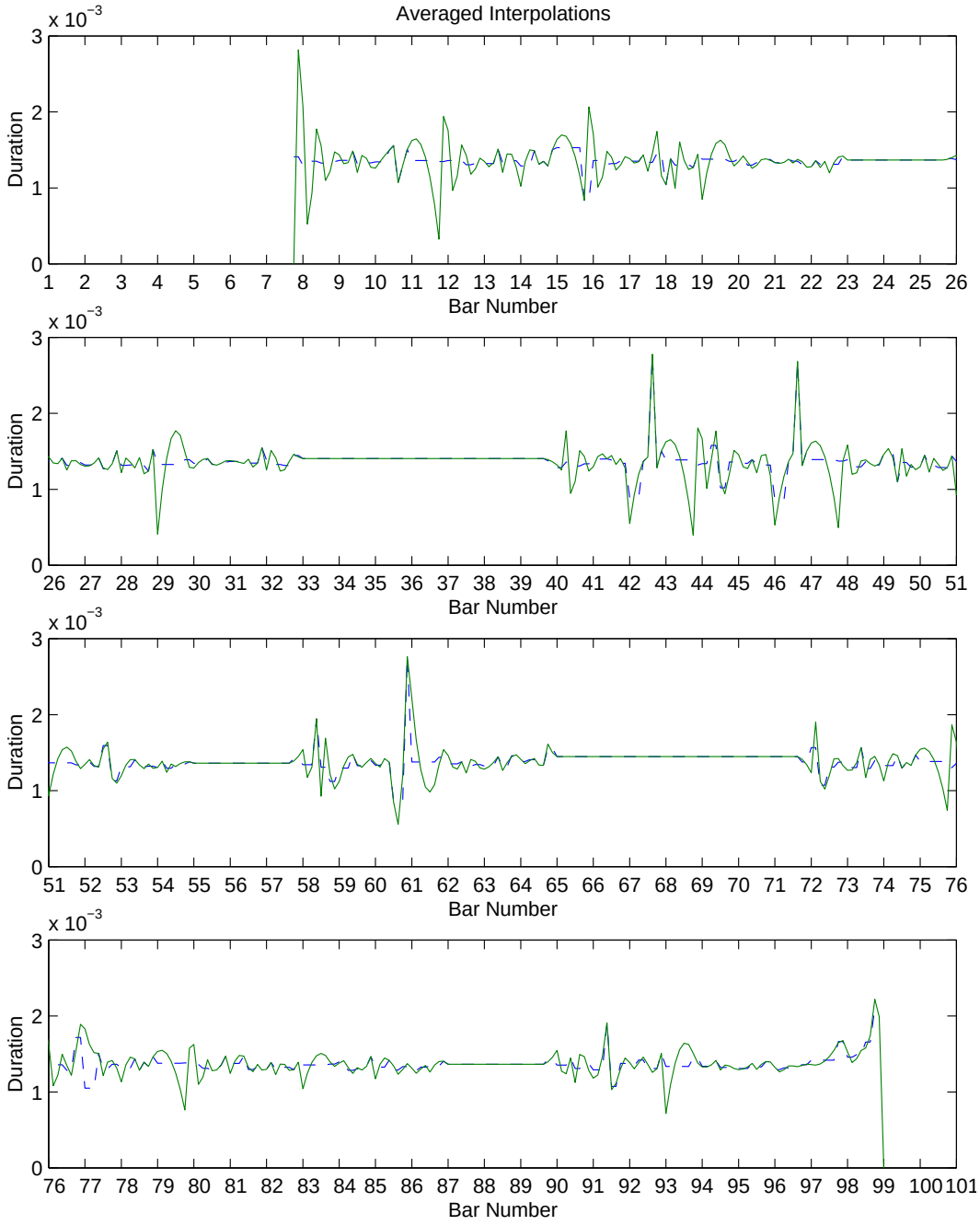


Figure 7.8: A graph showing the results of the simple interpolation (dashed-line) and context-based interpolated (solid-line) performance IOIs of *Gute Nacht*.

curve.

7.2.6 Summary

Overall, the results from the two different interpolation techniques are quite similar. The context-based interpolation technique performs well when there is enough context for it to base its decisions upon. However, when a series of points occur with little contextual information (such as long rests), the context-based interpolation produces some results that are not musically sensible.

The context-based interpolation does provide good results for the majority of the pieces and removes the step-wise changes in timing that are common with the simple method. For example, in Figure 7.6 the interpolation process produces a smooth series of curves for the timing deviations occurring between bars 27 and 31 whilst remaining close to the results of the simple interpolation process.

The context-based interpolation process introduces a number of interdependent equations which, although constrained by the information available, have a large degree of freedom. This freedom means that some artifacts are likely to arise and may affect the subsequent analyses.

Examining Figure 7.6 again, it can be seen that there is a large scale artifact introduced between bars 72 and 74. This artifact will affect the results of both of the analysis techniques. The autocorrelation analyses will show some false positives due to artificially high correlations whenever another curve is mapped onto the artifact due to the multiplicative nature of the analysis.

The curve-fitting analysis will also suffer under the artifact but instead of producing false positives may produce false negatives. For example, taking Figure 7.6, the oscillating feature during bars 72 and 74 will produce low curve-fitting results for any curve that spans across the two bars due to the large number of points which will inevitably lie far away from the fitted curve.

A method of interpolation is needed to provide a complete set of data for the analysis techniques used below. However, the curves produced by the simple interpolation process would be unsuitable due to the large flat areas. The context-based curves do provide a set of data suitable for analysis and do, for the most-part, provide a similarly

shaped curve to the simple interpolation.

7.3 Feature Identification

Viewing the performance data as a time series, there are a number of features which can be searched for. In the case of grouping structure, the features that are of most interest are those that can be obtained from the timing curve which identify a hierarchy of phrasal structures. Such features are:

- Repeating timing patterns – assuming that a recurring phrase is performed with a similar timing profile, then any instance of a repeating timing profile in the timing curve could denote a phrase repetition. This applies for both lower-level phrases (i.e. close to the musical surface) and higher-level structures.

- Parabolic timing patterns – any instances of *phrase-final lengthening* should result in a timing pattern that can be approximated by a parabola (Todd, 1989c; Repp, 1992a). Any contiguous areas of the timing curve that can be closely approximated by such a curve may correspond to an interesting structural unit.

To identify occurrences of these features, two analysis techniques are applied to the interpolated performances: autocorrelation and curve-fitting.

Autocorrelation is a standard statistical technique that has been shown to be successful when applied to musical performance data (Desain and Vos, 1990). However it does not provide the positional information required by the system presented in this thesis to identify boundaries. It is included here primarily to provide a comparison with the proposed curve-fitting analysis technique.

The novel curve-fitting analysis (described in detail in Section 7.3.2) tries to map occurrences of curves which could represent phrase-final lengthening onto the performance data. This curve-fitting process is conducted for all possible starting positions and curve lengths with the aim of finding patterns of curve-fits amongst the data which represent patterns of musical structure.

A future system could perhaps incorporate multiple techniques to identify the performance features. However, this research will focus on the effectiveness of only using

the novel curve-fitting approach and make use of autocorrelation as a comparative tool.

7.3.1 Autocorrelation

Autocorrelation is the process of comparing a time-series with itself with a certain lag. If the time series begins to repeat itself, or has some repetitive structure within it, then the autocorrelation process will return high correlations at those lags which correspond to the repetitive structure's length.

The formulae used to measure the autocorrelation are given in Equations 7.12–7.14, where $-1 \leq r_k \leq 1$ is the strength of the correlation at lag k.

$$c_k = \frac{1}{N}\sum_{t=1}^{N-k}(y_t - \bar{y})(y_{t+k} - \bar{y}) \tag{7.12}$$

$$c_0 = \frac{\sum_{t=1}^{N}(y_t - \bar{y})^2}{N} \tag{7.13}$$

$$r_k = \frac{c_k}{c_0} \tag{7.14}$$

where N is the number of points in the series, k is the lag, y_t is the value of the time series at position t and $\bar{y}$ is the mean value of y.

Figure 7.9 shows the result of applying autocorrelation to the example of timing structure given in Todd (1989a). The autocorrelation peaks at lags $k = 8$ and $k = 16$ indicate that there is a repetitive structure which occurs after 8 musical events and another which occurs after 16 musical events. In fact, the peak at $k = 16$ is expected because it is a harmonic of the repetitive structure at $k = 8$. However, there may also be a structure that is 16 events wide which also contributes to the correlation at this point. The peak at $k = 1$ can be discounted for the purposes of this analysis because it is due to the fact that many notes tend to be similar to their immediate neighbour and a repetitive structure one event wide makes little musical sense.

Another similar technique which can be applied to the same data is partial autocorrelation (Desain and Vos, 1990). Partial autocorrelation shows the amount of correlation between the time series and itself that is not accounted for by correlations at the smaller lags. For example if a repetitive structure occurs at lag P, then with the ordinary autocorrelation that structure will contribute to the correlation at lags 2P, 3P, 4P, etc. Partial autocorrelation corrects for this by removing the effect of smaller

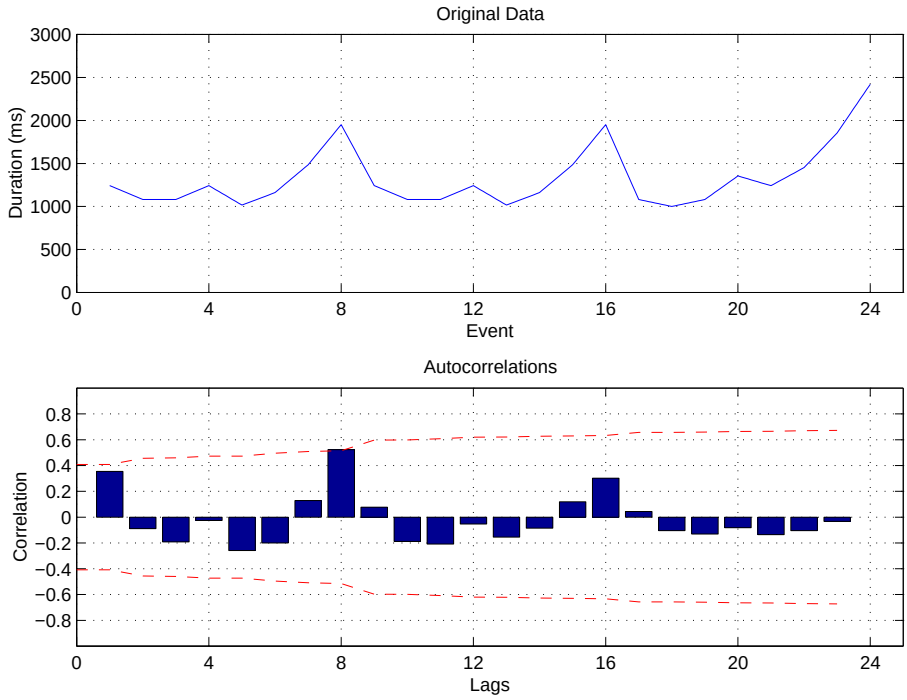


Figure 7.9: The top graph shows the original performance durations (adapted from Todd (1989a)). The bottom graph shows the results of applying autocorrelation. (Measures of significance ($p < 0.05$) are shown as dashed lines at $\pm 2\times$ standard error.)

lags from subsequent calculations.[4] Figure 7.10 shows the result of applying partial autocorrelation to the original data from Figure 7.9.

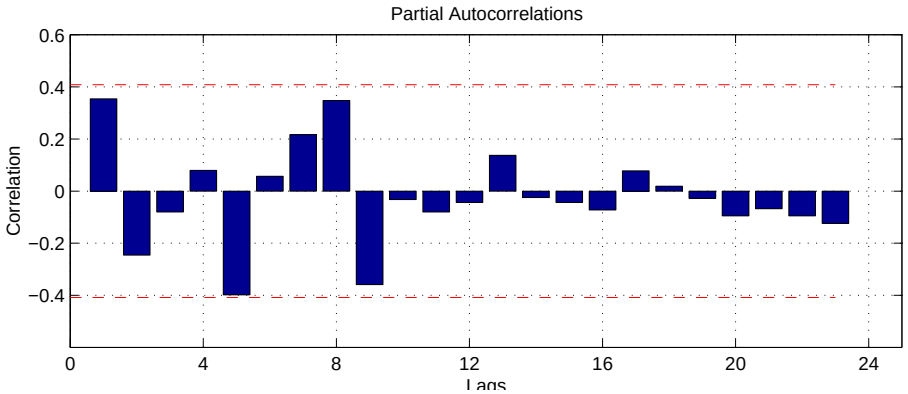


Figure 7.10: The results of applying partial autocorrelation to the original data shown in Figure 7.9. (Measures of significance ($p < 0.05$) are shown as dashed lines at $\pm 2/\sqrt{N}$.)

The large peak at $k = 8$ still remains, however the second peak at $k = 16$ has considerably diminished in size. As before the peak at $k = 1$ can be ignored and although the peak at $k = 7$ is also large, because it neighbours the larger peak of $k = 8$, it too can be ignored due to the fact it is probably an indication that the curve is approaching a lag when it will be in phase again.

Using both these autocorrelation methods, any repeating patterns in the timing curves will be identified. However, even if such patterns are identified, the processes do not identify where those patterns occur, only that they exist.

7.3.2 Curve Fitting

As mentioned above, another technique to identify musical structure is to try to find timing curves that suggest musical phrases. This curve fitting process will concentrate on locating sections of the performance where the shape of the timing curve can be modelled closely by the repeated application of a quadratic curve.

[4] More details on partial autocorrelation can be found in Appendix C or Box and Jenkins (1976).

To identify these features in the performance, a sliding window technique is used. For each point x in the performance, a window size w is chosen and the error of the curve that best fits the data in the range $[x, x+w)$ is recorded. The start of the window is then placed at $x+w$ in the data and the error recorded again. This process is repeated until the end of the performance has been reached, the average error is then recorded for the starting position x and window size w. The average error from the curve-fitting is recorded for all possible starting positions and window sizes.

To find the best quadratic curve that fits the data in the window, a least-squares approximation is used as described below.

7.3.2.1 Least-squares Approximation

This section describes how a least-squares approximation is used to find the curve that best fits a set of data.

Given a set of m data points:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) \tag{7.15}$$

where the y_i are known to be subject to errors. The goal is to find a function $F(x)$ such that:

$$y_i = F(x_i) + \eta_i \tag{7.16}$$

for $i = 1, 2, \ldots, m$, where the error terms η_i are as small as possible. In the case of a perfect fit the η_i terms would all be zero. F is chosen to be a linearly weighted sum:

$$F(x) = \sum_{j=1}^{n} c_j f_j(x) \tag{7.17}$$

where n and the basis functions f_i are chosen to suit the phrase-final lengthening model of performance. Since the model uses quadratic functions to explain the timing curves, setting $f_j(x) = x^{j-1}$ and $n = 3$ gives:

$$F(x) = c_1 + c_2 x + c_3 x^2 \tag{7.18}$$

Let the matrix A denote the values of the basis functions at the given data points:

$$A = \begin{pmatrix} f_1(x_1) & f_2(x_1) & f_3(x_1) \\ f_1(x_2) & f_2(x_2) & f_3(x_2) \\ \vdots & \vdots & \vdots \\ f_1(x_m) & f_2(x_m) & f_3(x_m) \end{pmatrix} \tag{7.19}$$

Let the column matrix c denote the desired matrix of coefficients:

$$c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \tag{7.20}$$

Then:

$$Ac = \begin{pmatrix} f_1(x_1) & f_2(x_1) & f_3(x_1) \\ f_1(x_2) & f_2(x_2) & f_3(x_2) \\ \vdots & \vdots & \vdots \\ f_1(x_m) & f_2(x_m) & f_3(x_m) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} F(x_1) \\ F(x_2) \\ \vdots \\ F(x_m) \end{pmatrix} \tag{7.21}$$

where the column matrix Ac consists of the m values which would be obtained by applying the function F to each value of x recorded. The corresponding column matrix η of errors is given by:

$$\eta = Ac - y \tag{7.22}$$

where y is the column matrix of the measured values for each x.

In order to minimize the errors, the norm of the error matrix η must be minimized. This is the equivalent of a least-squares approximation since:

$$\| \eta \| = \left(\sum_{i=1}^{m} \eta_i^2 \right)^{1/2} \tag{7.23}$$

Now:

$$\| \eta \|^2 = \| Ac - y \|^2 = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} c_j - y_i \right)^2 \tag{7.24}$$

The term $\| \eta \|$ can be minimized by differentiating $\| \eta \|^2$ with respect to each c_k and setting the result to 0.

$$\frac{d \| \eta \|^2}{dc_k} = \sum_{i=1}^{m} 2 \left(\sum_{j=1}^{n} a_{ij} c_j - y_i \right) a_{ik} = 0 \tag{7.25}$$

This is equivalent to:

$$\begin{aligned} (Ac - y)^T A &= 0 \\ A^T (Ac - y) &= 0 \\ A^T Ac &= A^T y \end{aligned} \tag{7.26}$$

which finally gives:

$$c = ((A^T A)^{-1} A^T) y \tag{7.27}$$

A solution to Equation 7.27 will give values to the coefficients c_i which describe the best curve that fits the original m data points.

7.3.2.2 Applying Least-Squares Approximation

The error is then measured between the curve corresponding to the best fit over the data points and the data points themselves. The error is defined in Equation 7.28, where $\eta_{x,w}$ is the error, x is the starting point of the window, y_i is the recorded data at point x_i, w is the window size and F is the function found by the least-squares approximation.

$$\eta_{x,w} = \sum_{i=x}^{x+w} (y_i - F(x_i))^2 \tag{7.28}$$

As described above, the goal of this process is to find sequences of points that lie along smooth curves. If such a sequence of points exists, it will correspond to a series of curve-fits over the data that results in a relatively low measurement of error.

Algorithm 7.1 shows pseudo-code for how the total error for each possible value of x and w is calculated. There will be approximately $N^2/2$ measurements recorded where N is the number of possible starting points. The final error recorded is the average error of all the repeating windows starting at x with size w.

Algorithm 7.1 Moving variable-sized repeating-window curve-fitting algorithm

PHRASEFINDER()

```
1: for x ← 1 to N do
2:    for windowsize ← 1 to N − x do
3:       startx ← x
4:       error ← 0
5:       windowcount ← 0
6:       while startx < N do
7:          error ← error+ CURVEFIT(startx, windowsize)
8:          startx ← startx + windowsize
9:          windowcount ← windowcount + 1
10:      end while
11:      error ← error/windowcount
12:   end for
13: end for
```

Although the context-based interpolation process uses quadratic curves to interpolate the missing points, it is important to note that the resulting curves used for the interpolation process could be either convex or concave. The context around the interpolated points would dictate the sign of the x^2 coefficient. However, for the curve-fitting algorithm, only the best fitting curves with a non-negative x^2 coefficient will be viewed as good fits in line with Todd's model. Curves that have a negative x^2 coefficient (i.e. concave) will be penalised with a large fixed error whereas curves that have non-negative x^2 coefficients will receive an error value proportional to how closely they fit the data.

An example of applying curve-fitting: Figure 7.11 illustrates some interesting aspects of the curve fitting process. The top-most graph shows the raw performance data of a piece in 4/4 time. The second graph shows some example best fit curves that have been applied starting at beats $4, 8, 10, 14$ and 18 with a window size of 4 units. The starting points are used to illustrate the large range of curves that will fit even this small example. For example, the curves at 4 and 8 are good matches for the phrase-

final lengthening model. The curves at 10, 14 and 18, although fitting the data quite well does not provide a good match for the phrase-final lengthening model.

The first two of these curves have positive x^2 coefficients; the last three have negative x^2 coefficients.

The third graph in Figure 7.11 shows the result of plotting error against starting position for windows of size 4 units. The lowest errors occur whenever the selected window encompasses an entire peak or trough.

However, if the model of phrase-final lengthening is being used to find phrase boundaries, only the occurrences of best curve fits that have non-negative x^2 coefficients are of interest.

The bottom-most graph in Figure 7.11 shows the error if only non-negative x^2 coefficients are considered. In this example, if the x^2 coefficient of the best fit curve was negative, the error was assigned a constant representing the maximum error previously recorded.

Examining this graph, it can be seen that the lowest errors occur at the start and end of the phrase boundaries which are being sought. More correctly, the lowest errors occur at the start and end of what are assumed to be phrase boundaries due to the shape of the timing curve. For example, from beats 8–16 there are three troughs; one at beat 8, one at beat 12 and one at beat 16. When these three troughs are compared with the timing curve in the top graph, it is clear that they correspond to the starts and ends of two parabolic curves.

Figure 7.12 shows a similar set of graphs, but with a window size of 8 units. The bottom-most graph shows the result of applying the curve-fitting whilst penalising those curves that have non-positive x^2 coefficients. As can be seen from the graph, the troughs at $x = 8$ and $x = 16$ identify the high-level phrase structures that were not identified by using a window size of four. The curve also does not have troughs at $x = 4$ or $x = 12$ which correspond to the locations of the smaller four-beat wide structures identified earlier.

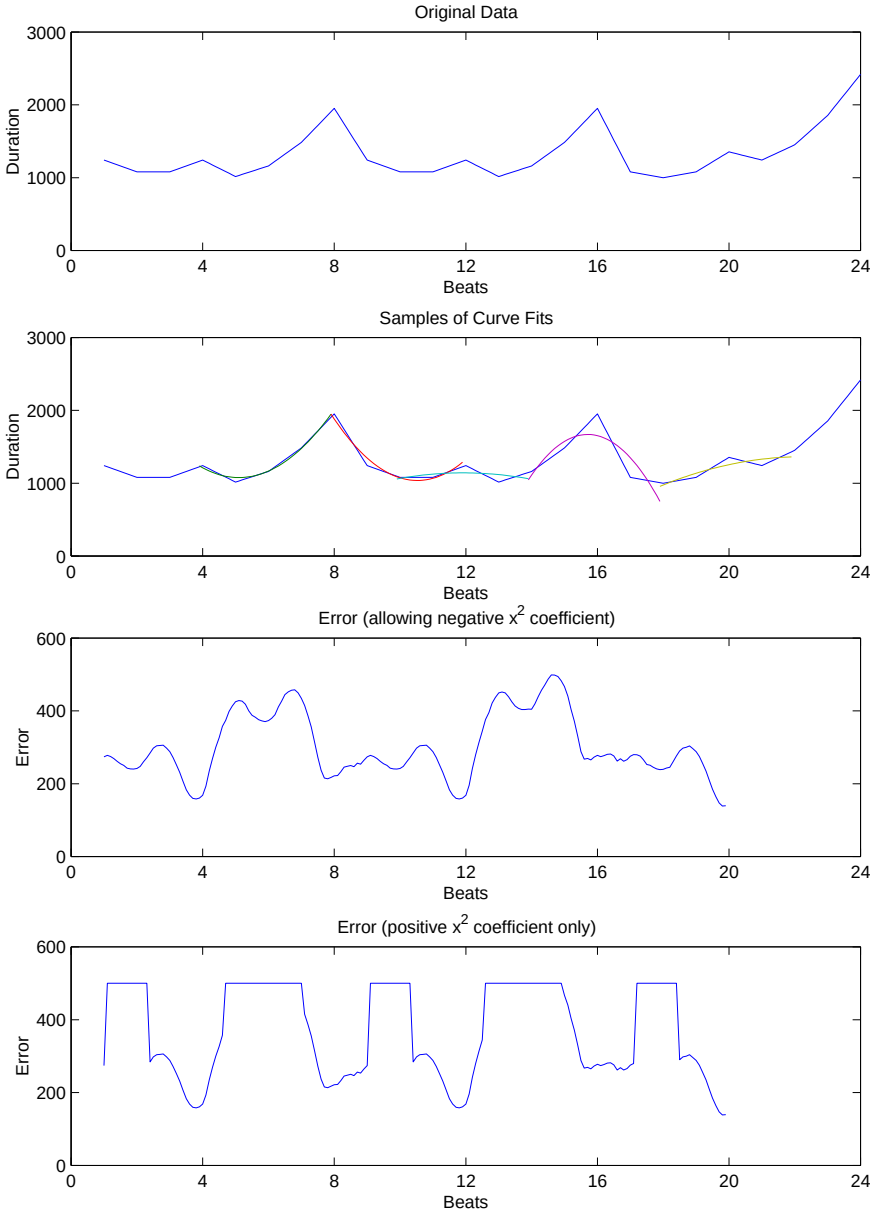


Figure 7.11: Curve fitting example. The topmost graph shows the original performance data, the second graph illustrates some possible quadratic curves (of length four) that can be fitted to the data. The third graph shows the resulting errors from these curves. The final graph shows a similar set of errors, but with those curves with a negative x^2 coefficient penalised.

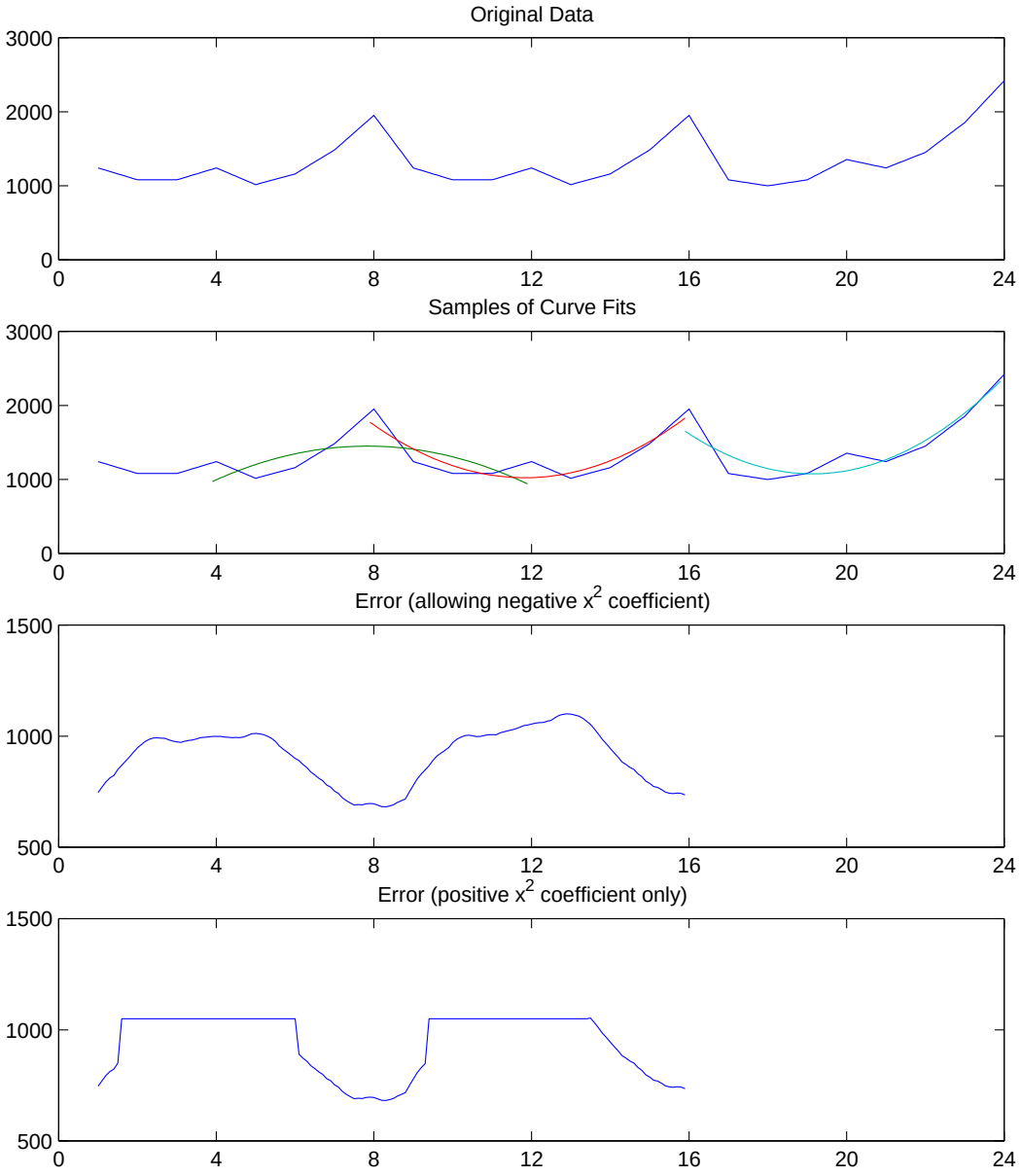


Figure 7.12: Curve fitting example. The topmost graph shows the original performance data, the second graph illustrates some possible quadratic curves (of length eight) that can be fitted to the data. The third graph shows the resulting errors from these curves. The final graph shows a similar set of errors, but with those curves with a negative x^2 coefficient penalised.

7.4 Analysis

The previous sections introduced the two analysis techniques which are going to be applied to the interpolated data. The following sections describe the results of applying the analyses to the three sets of performances. Each performance was analysed as a whole and when divided into musically sensible sections (decided from the score).

The following sections present the results from the two analyses and examines if the same features that are detected by the autocorrelation analysis are detected by the curve-fitting analysis.

Importantly, this chapter only presents the results from the curve-fitting analysis whereas Chapter 8 describes how they can be interpreted.

7.4.1 Analysis: *Berceuse*

The following sections describe the results of applying the two analysis techniques to the interpolated performance of *Berceuse*.

7.4.1.1 Predicted Results

Berceuse has three main phases: the first phase has a repeating four-bar phrase, the middle part contains a transitionary theme and the final phase consists primarily of a one-bar phrase apart from one performance of the four-bar phrase from the first part.

The autocorrelation analyses of the whole piece should identify both the one-bar and four-bar repetitions, however their strength will be diminished due to their presence in only a third of the piece. The autocorrelation of the segments should strongly identify the four-bar structure in the first section and the one-bar phrase structure in the final section.

The curve-fitting process should identify the same structures as the autocorrelation, plus identify some of the other features such as the single occurrence of the four-bar phrase in the third part.

7.4.1.2 Autocorrelation Results

Figure 7.13 presents the results of applying autocorrelation and partial autocorrelation to the whole of the interpolated performance of *Berceuse*. The results of the autocorrelation show significant peaks at lags $k = 1, 9, 15, 24$ and lesser peaks for $k = 2, 3, 4, 6$ and $k = 10$.

The peaks at $k = 9,\ 15$ and $k = 24$ are explained by examining the last third of the performance duration graph. There are three peaks at bars 58, 73 and 82; the differences between these bar numbers being 15, 9 and 24 which corresponds with the lag values that produce the strong correlations. Of these three peaks, the two peaks at bars 58 and 82 correspond to valid interpolations of the performance data. However, the peak at bar 73 is an artifact of the interpolation process. Excluding the correlations that result from this peak leave the one large correlation of 24 which matches the duration of the last third of the piece.

The other large peak in the autocorrelation graph occurs at $k = 1$. The high correlation at the one-bar lag is expected given the musical structure of the last part of the piece which consists of the one-bar long repeating accompaniment.

The lesser peaks at $k = 2$ and $k = 4$ are expected given the four-bar structure of the main theme. The peaks at $k = 3, 6$ and $k = 10$ are not expected but may arise as harmonics of the peaks with smaller lags.

With this in mind, partial autocorrelation is applied to the same data to give the graph at the bottom of Figure 7.13. As described before, partial autocorrelation will remove any peaks due to repeating structures that have a smaller lag than the current lag.

The peaks of the partial autocorrelation occur at $k = 1, 9, 15$ and $k = 24$. As before, the peak at $k = 1$ is expected with the one-bar repeating phrase at the end of the piece. The peaks at $k = 9,\ 15$ and $k = 24$ are explained as above. Of more interest are the smaller peaks that remain at $k = 2,\ 4,\ 6,\ 12$ and $k = 19$. The strong correlation at $k = 4$ is desired due to the four-bar phrase structure at the beginning of the piece. The other peaks at $k = 2,\ 6$ and $k = 12$ are less readily explained but could be due to a higher level repetitive structure in the piece.

The peak at $k = 19$ cannot be readily explained from the score and is unusual given

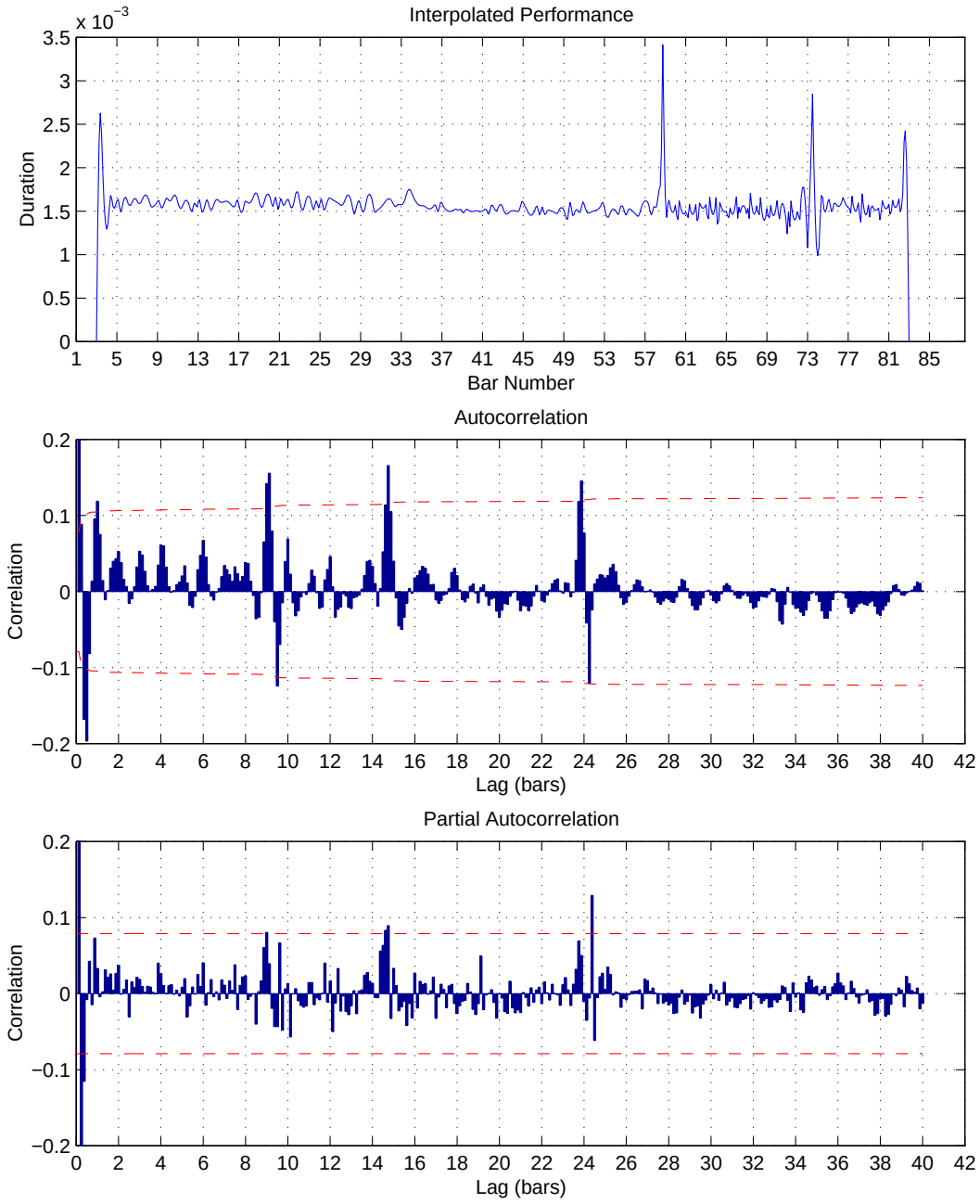


Figure 7.13: A figure showing the interpolated performance of *Berceuse* (top), the results of applying autocorrelation (middle) and the results of the partial autocorrelation method (bottom).

that it is not present in the autocorrelation results.

A possible reason for the spurious results is the fact that the piece has three distinct phases, each of which has its own properties and phrase structure. The interaction between these three phases could be the source of these artifacts.

Figure 7.14 shows the results of applying both the autocorrelations to the piece as three distinct sections. The first row of graphs corresponds to the first 34 bars of the piece during which the main four-bar theme is repeated. The second row is based on the 24 bar transitional period ending when the primary and secondary voices swap rôles. Finally, the third row corresponds to the last third of the piece during which the primary voice has become the accompaniment to the second.

Taking each row of the figure in turn; the results of the autocorrelation of the first phase show that there are strong peaks at $k = 1$, $k = 4$ and at $k = 15$. The first two of these peaks are what are expected given the four-bar phrase structure. The peak at $k = 15$ is not readily explainable from the score except for the fact that it is approximately half the size of this first section of the piece and it corresponds to a point in the performance where the piece has swelled to a crescendo and is returning to a *piano* section.

Examining the partial autocorrelation shows the same peak at $k = 4$ (although smaller) and $k = 15$. There is also a new peak at $k = 2$, which does seems plausible given the four-bar phrase structure.

The second row of Figure 7.14 shows the transitionary section of the piece. The autocorrelation shows strong correlations at $k = 2,\ 6$ and $k = 8$. During this part of the piece, the phrase structure is less obvious from the score than the first and the last sections. However, upon listening to the performances, the structure suggested by the two-bar lag is apparent. The peaks at $k = 6$ and $k = 8$ are less obvious but again may correspond to a higher level structural feature. The partial autocorrelation for this second sections offers a similar set of peaks.

Finally, the third row of the figure shows the results from applying the autocorrelation processes to the last section of the piece. This section of the piece primarily consists of a one-bar phrase accompanying the second voice which has now become the lead. The autocorrelation analysis provides strong peaks at $k = 1, 3, 6, 9$ and another

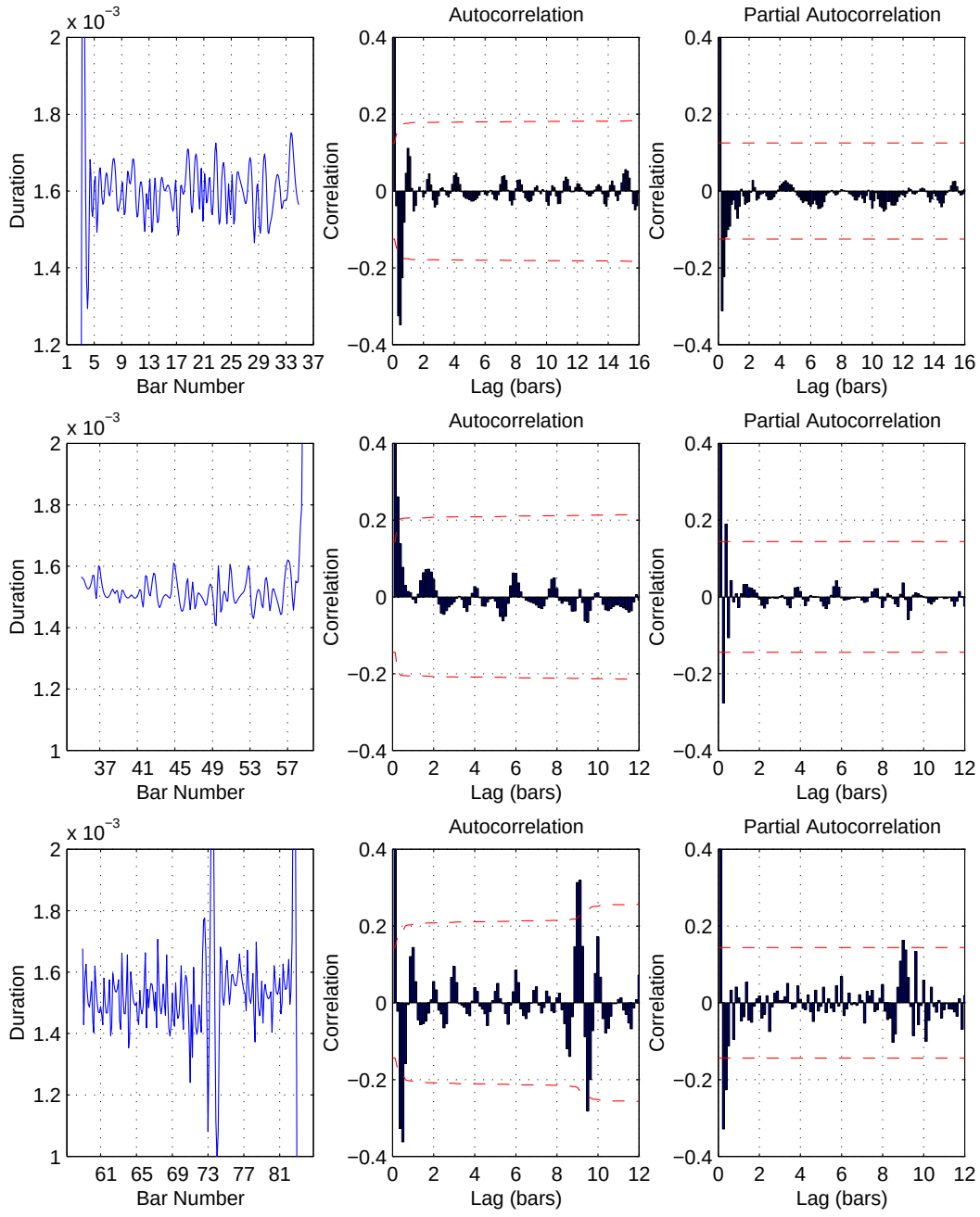


Figure 7.14: A figure showing the results of applying the autocorrelation and partial autocorrelation processes to the three constituent parts of *Berceuse*.

at $k = 10$. The peak at $k = 1$ is expected given the one-bar repeating accompaniment, however the peaks at multiples of three are unexpected given that the first voice is accompanying a similar four-bar phrase as occurred at the start of the piece. The peak at $k = 9$ corresponds to the lag which causes the last two peaks, at bars 73 and 82, in the interpolated performance to overlap.

The partial autocorrelation is less conclusive, offering a large number of weak peaks, however the peak at $k = 9$ remains strong.

7.4.1.3 Curve-Fitting Results

Figure 7.15 shows the results of the curve-fitting process when applied to the interpolated performances of *Berceuse*. The figure shows the results of applying the repeating curve-fitting algorithm to the performance with varying starting positions (x-axis) and window sizes (y-axis). The error between the best fit curve and the actual data is plotted on the graphs with blue indicating a low error (i.e. a good fit) and red indicating a high error (i.e. a poor fit). Occurrences of low errors which indicate that a convex quadratic curve was able to fit closely to the performance data are of special interest in this research.

Under the assumption that the music primarily consists of regular phrase structures, an individual point of low error is relatively unimportant and will not be considered a clue to the musical structure. However, contiguous areas of low error or patterns of alternating weak and strong errors suggest features that arise due to the presence of such a regular phrase structure.

In the following sections, three types of features are discussed:

- Vertical patterns of low-error which suggest the start of a phrase boundary;
- Diagonal stripes of low-error which suggest the end of a phrase boundary;
- Horizontal patterns of alternating strong and weak curve-fits which indicate a regularly occuring phrase structure.

Section 8.2 contains a more detailed explanation of each these features with the aim of automatically detecting them.

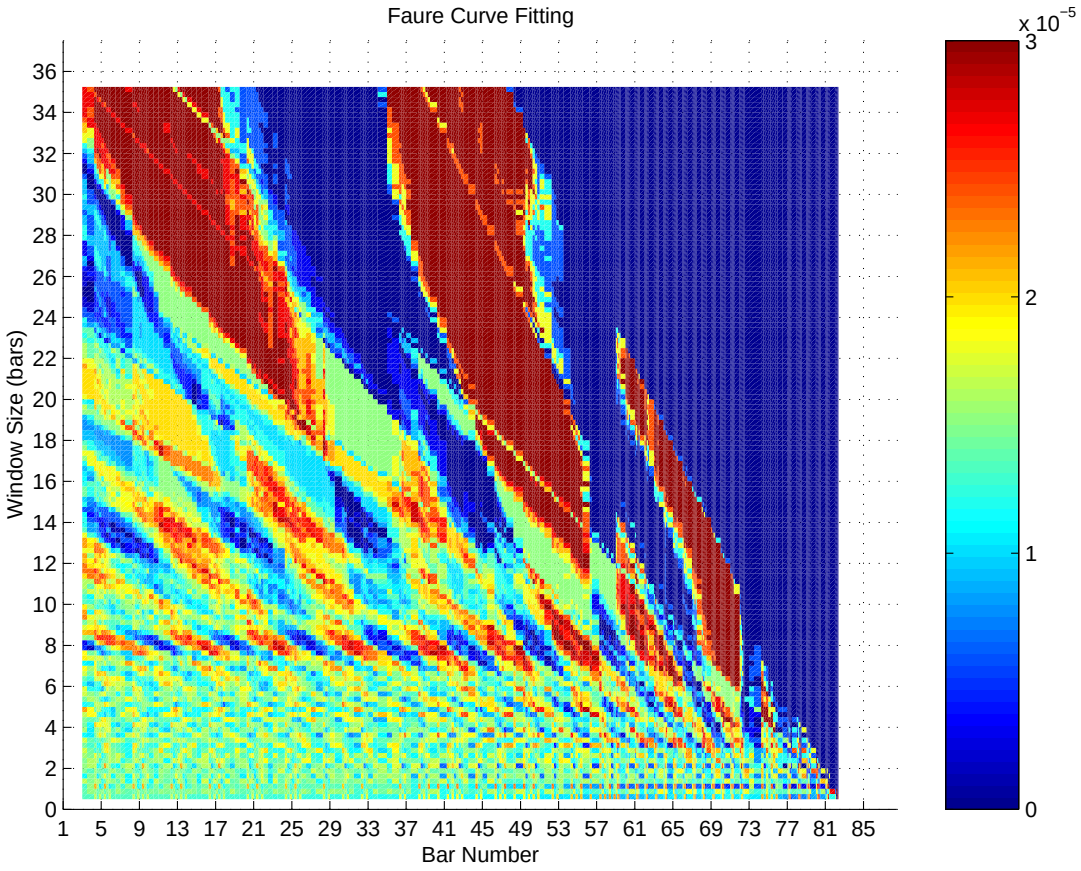


Figure 7.15: Curve fitting results for *Berceuse*.

It is clear from the figure that there are a large number of horizontal patterns of alternating weak/strong errors, e.g. at $y = 2$ or $y = 8$, which suggests that a repetitive structure may be present in the musical structure. However, as with the autocorrelation analysis, the curve-fitting analysis of the piece as a whole is too crude and is subject to potentially conflicting data from the differing sections of the piece. Therefore, as with the autocorrelation results, each section of the piece will be examined in turn to try to identify the features that are only applicable to that section of the piece.

Figure 7.16 shows the results of the curve fitting process for the first third of the piece. There is mix of high and low error results.

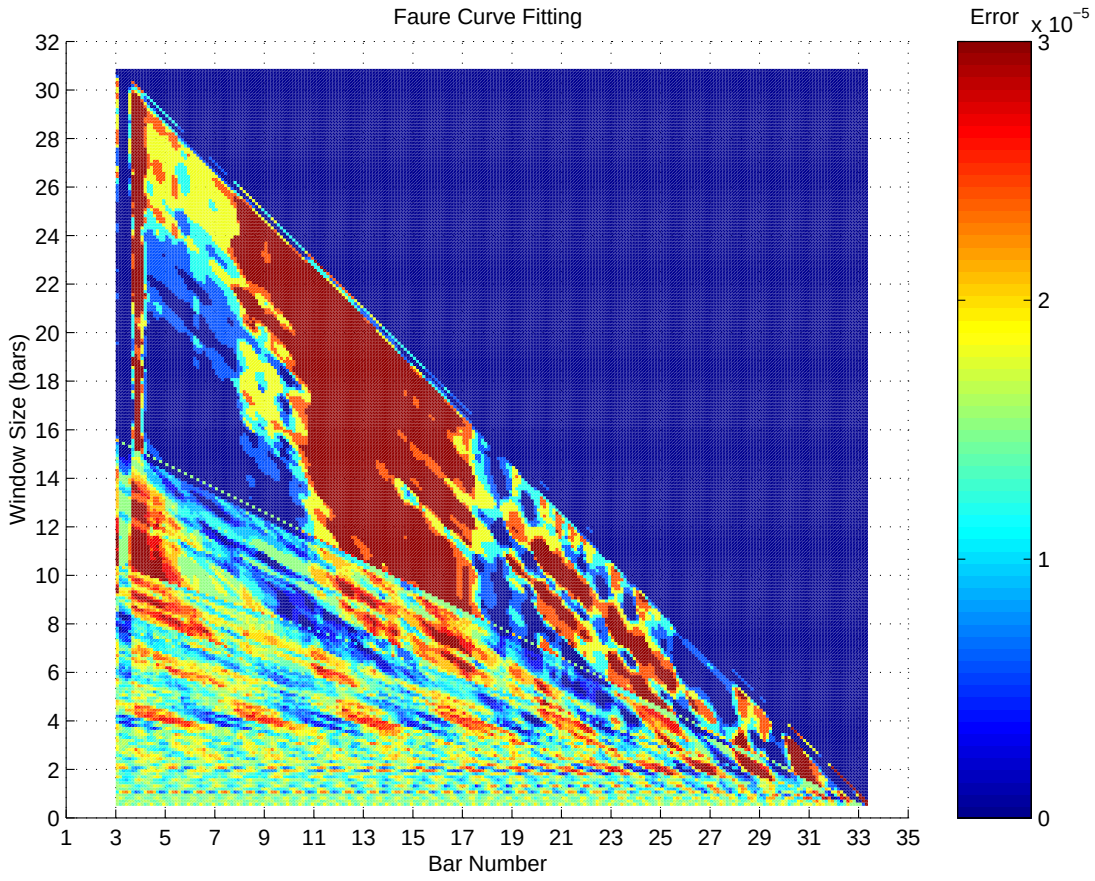


Figure 7.16: Detail of curve fitting results for the first third (up to bar 34) of *Berceuse*.

There are two large features that dominate the top half of the graph, the area of low errors (based around $(6, 17)$) and the area of high-errors (based around $(13, 16)$).

These indicate that there is a set of relatively large curves that, when begun in the first third of the piece, can provide a good approximation for a large section of the piece. These sets of curves correspond to the shape of the performance data which suggests a change in the average duration between bars 5–17 and bars 19–25.[5]

There are also distinct features present in the lower bottom half of the graph. For y values of 1, 2, 4 and, less clearly, $y = 8$ there are horizontal patterns of alternating weak/strong error values recorded. These agree with the results from the autocorrelation analysis which detected peaks for similar window sizes.

Figure 7.17 shows the results of the curve fitting process when applied to the middle section of *Berceuse*. In contrast to the results from the first section of the piece, there are few large scale features in the results. There is a horizontal pattern of alternating error strengths at $y = 2$. There are some vertical features of low errors at $x = 35, 42$ and $x = 44$. From $y = 18$ to $x = 53$ and from $y = 22$ to $x = 57$ there are diagonal stripes of low error measurements.

Finally, Figure 7.18 shows the curve fitting results for the final third of the piece. The most prominent feature in this figure is the large area of low error around $(62, 14)$. There is also a diagonal stripe of low error terminating at $x = 73$ along with two vertical features either side of $x = 73$. A strong horizontal pattern of alternating error strengths occurs at $y = 1$ with slightly less distinct patterns occurring at $y = 0.5, 1.5$ and $y = 2$.

7.4.2 Analysis: *Auf dem Hügel sitz ich spähend*

The same analysis techniques as above were applied to *Auf dem Hügel sitz ich spähend*. In this piece, there are five stanzas separated by relatively long rests. Each analysis technique is again applied to the whole performance and to each of the five stanzas in turn.

7.4.2.1 Predicted Results

The piece consists of five sections – corresponding to the five stanzas of the poem it was composed for. Each section is short and only lasts 8 bars with a two-bar rest

[5] This is evident both from visual inspection and when a moving average is applied across the performance data.

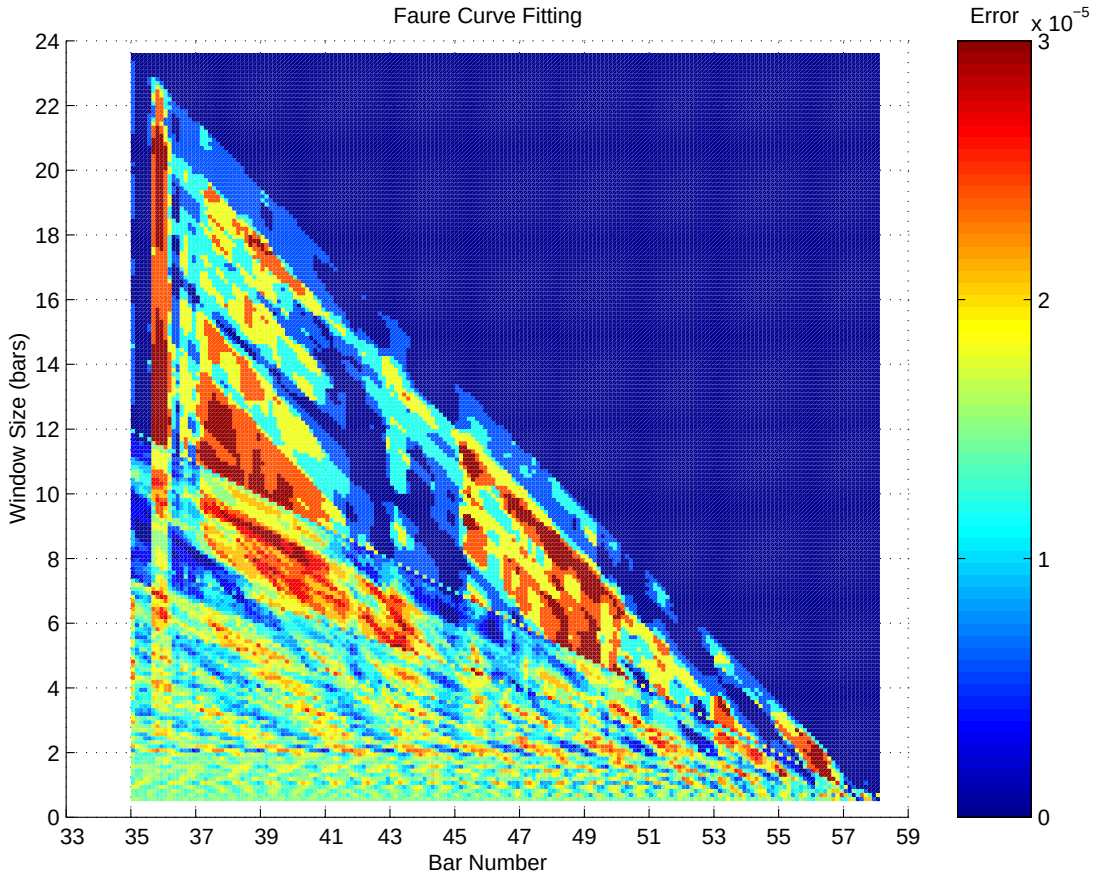


Figure 7.17: Detail of curve fitting results for the middle third (bars 35 to 58) of *Berceuse*.

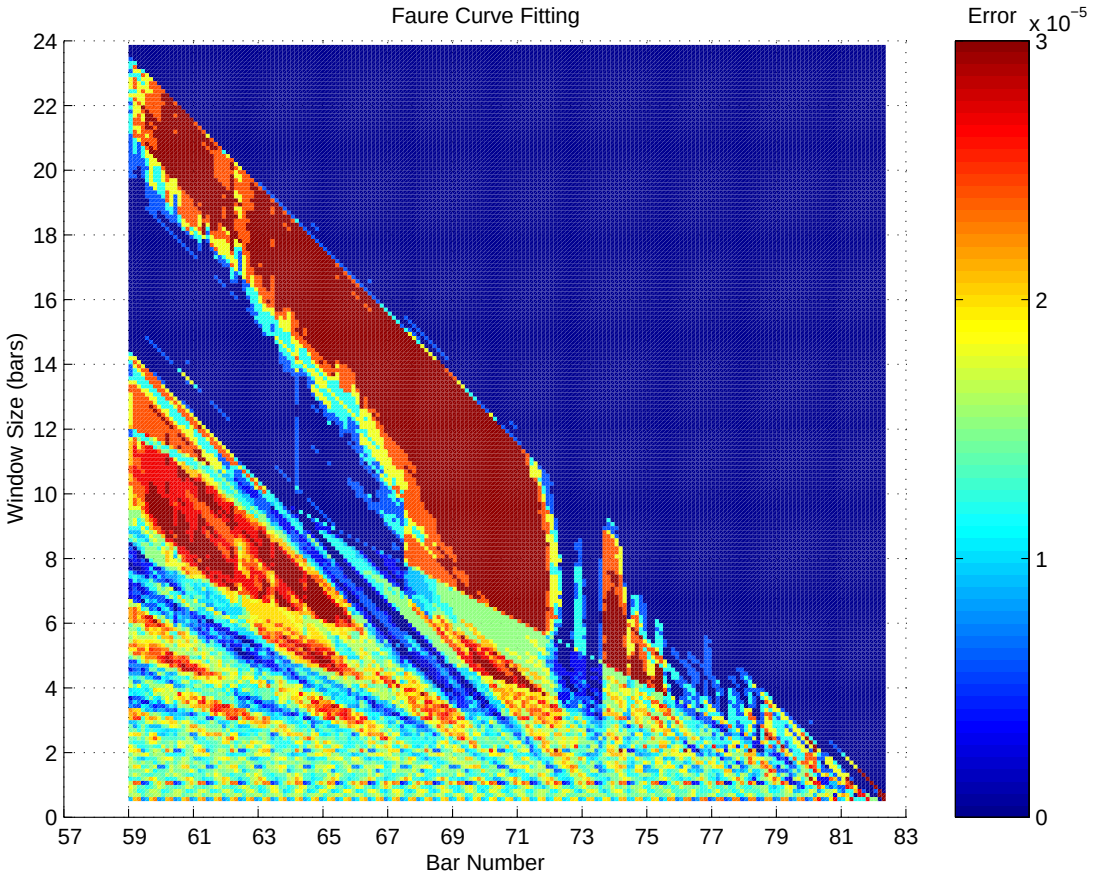


Figure 7.18: Detail of curve fitting results for the final third (from bar 59) of *Berceuse*.

between them. The section consists of a two-bar phrase followed by a six-bar phrase. The six-bar phrase consists of three parts each of approximately two bars.

With the autocorrelation results, a strong correlation around the two-bar lag is expected. Beyond that, at a higher level, a correlation of 10 bars should be present to account for each stanza.

The curve-fitting process should strongly identify each section and the phrasal structure within them.

7.4.2.2 Autocorrelations

Figure 7.19 shows the results of applying the autocorrelation processes to the performance data. The strongest peaks in the autocorrelation results occur around $k = 1, 3, 9, 10.5, 12$ and $k = 17$. These values do not agree with the phrase structure (which would suggest the presence of a two-bar feature). The peak at $k = 10.5$ may be the expected peak at 10-bars offset due to the contribution of other factors. The large peaks at $k = 9, 12$ and $k = 17$ can be explained by the interactions of the large peaks in the performance data at $x = 6, 12, 22, 29, 35, 38$ and $x = 46$.

The partial autocorrelation results are less-clear than the autocorrelation values. Again, a similar set of peaks as for the autocorrelation results arise, however new peaks at $k = 6$ and $k = 22$ have been formed. The peak at $k = 22$ seems to be an artifact as it suggests a correlation that is larger than the overall stanza based-structure of the song. The peak at $k = 6$ may however be representative of the high-level structure within each stanza.

Figures 7.20 and 7.21 show the results of applying the autocorrelation processes to each of the five sections of the piece in turn. The results are varied despite the common structure of all five parts. This may be due to the relatively short length of each section.

The first section shows a number of weak correlations around the $k = 1$ and $k = 3$ lags with the lag at $k = 3$ replaced by the more expected lag of $k = 2$ when partial autocorrelation is applied. The second section shows only one significant lag, that of $k = 2.5$ which occurs in the autocorrelation and is replace by a peak at $k = 2$ in the partial autocorrelation results. The third section shows a strong correlation at lags $k = 1$ and $k = 2.5$ for both the autocorrelation and partial autocorrelation results.

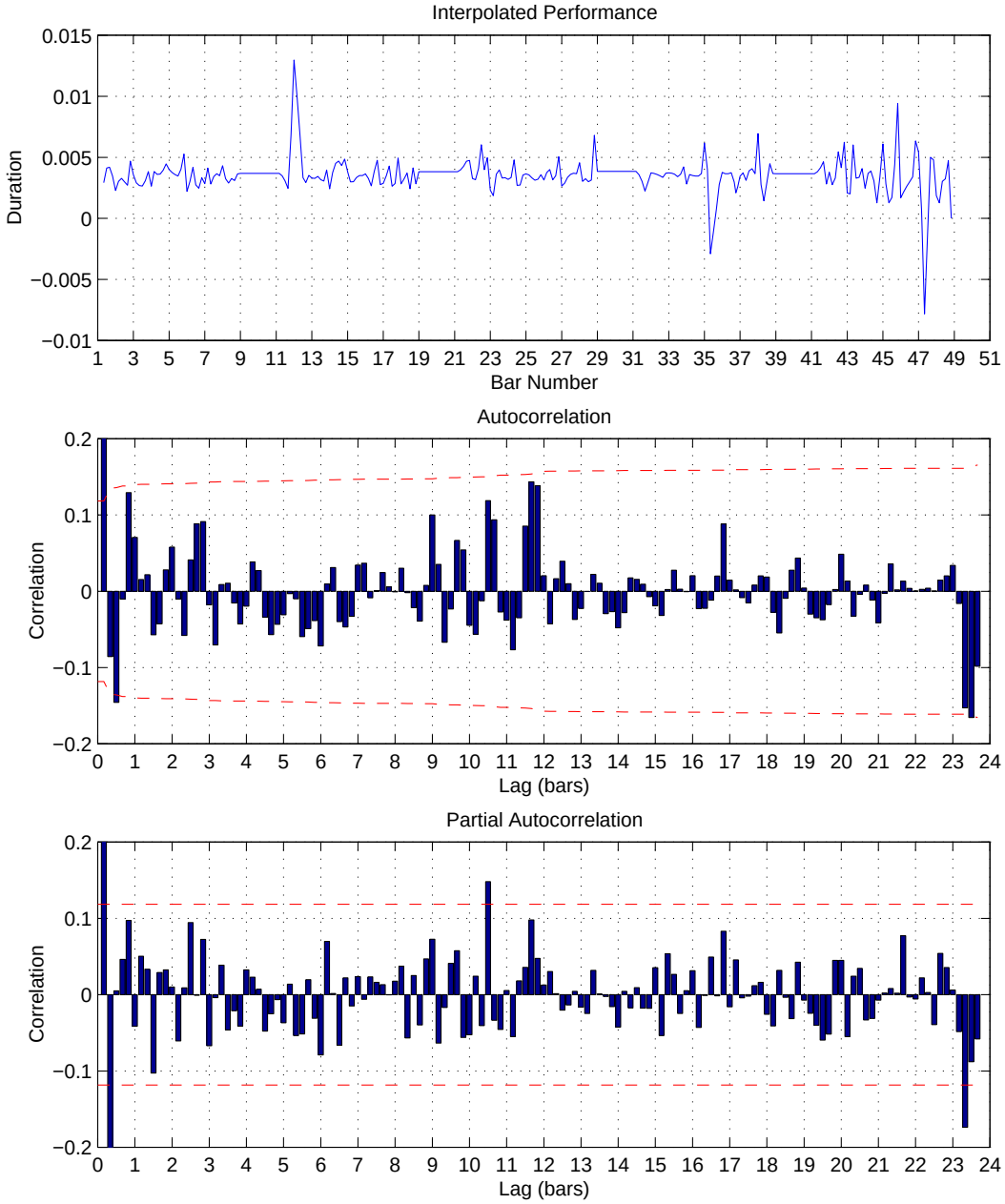


Figure 7.19: A graph showing the results of applying autocorrelation and partial autocorrelation to the whole of *Auf dem Hügel sitz ich spähend*.

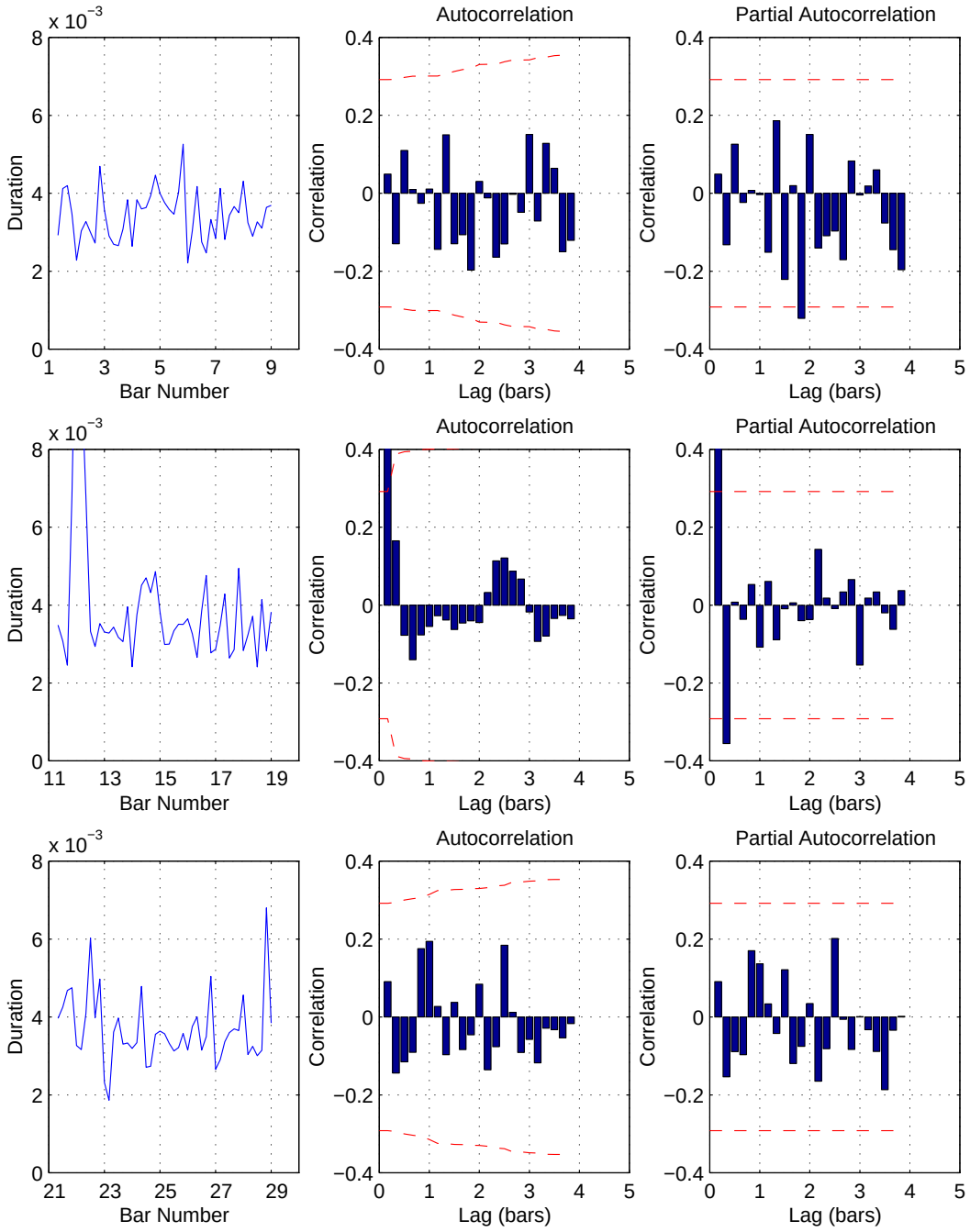


Figure 7.20: A graph showing the results of applying autocorrelation and partial autocorrelation to the first three sections of *Auf dem Hügel sitz ich spähend*.

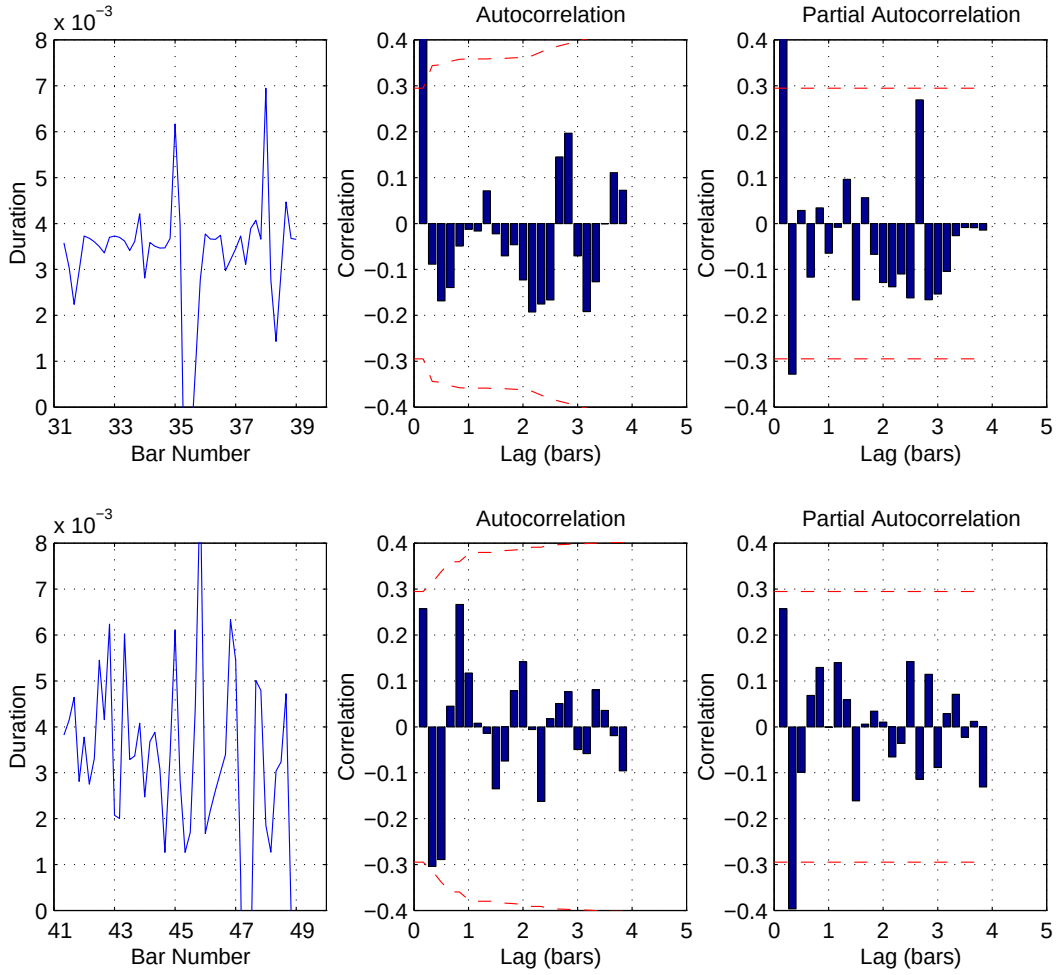


Figure 7.21: A graph showing the results of applying autocorrelation and partial autocorrelation to the last two sections of *Auf dem Hügel sitz ich spähend*.

The results for the last two sections show a similar mix of values. The fourth section provides strong evidence towards a three-bar repeating structure whereas the autocorrelation results of the fifth section suggest lags of $k = 1, 2$ and $k = 2.5$.

7.4.2.3 Curve Fitting

Figure 7.22 presents the results of applying the curve-fitting process to the whole of *Auf dem Hügel sitz ich spähend*. When compared with the results from the *Berceuse* analysis, the most obvious difference is the scarcity of areas of low error. It is difficult to distinguish any clear patterns from these initial results. At $y = 3$ there is a horizontal pattern of alternating moderate to strong errors. Similarly at $y = 11$ there is a suggestion of another horizontal pattern.

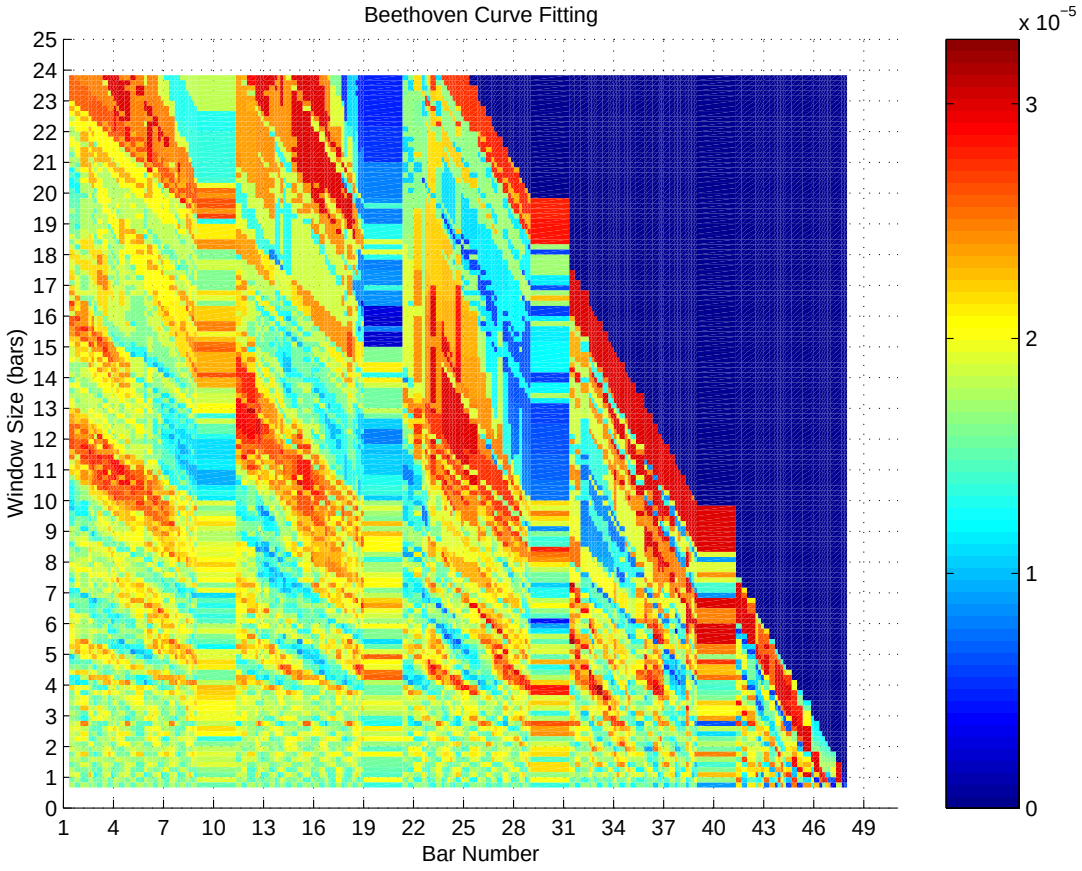


Figure 7.22: Curve fitting results for *Auf dem Hügel sitz ich spähend*.

Figure 7.23 shows the results of applying the curve-fitting process to each section of *Auf dem Hügel sitz ich spähend*. Again, the results for each of the sections contain less patterns than the *Berceuse* sections. The first section contains some weak vertical stripes of low errors at $x = 1, 3, 4$ and $x = 6$. The second section displays some weak vertical stripes at $x = 11.5$ and $x = 15$. Similarly to the first two sections, the third section only contains vertical features. In this section, the vertical features are at $x = 21.5$ and $x = 22.5$.

The fourth and fifth sections present patterns of features which are more reminiscent of the *Berceuse* results. In the fourth section, there is a horizontal pattern of alternating weak to moderately strong error measurements at $y = 1$. This is complemented by two diagonal stripes ending at $x = 35$ and $x = 37.5$. The results from the final section contain two diagonal stripes; one terminates at $x = 45.5$ the other at $x = 46.5$.

7.4.3 Analysis: *Gute Nacht*

7.4.3.1 Predicted Results

The piece consists of three sections (27 bars, 27 bars, 29 bars) divided by long rests of 5 bars. Each section starts on the last quaver beat of the bar in which it begins. Each of the sections is itself divided into 6 phrases (corresponding to two verses from the poem). The sections consist of 4 phrases followed by a long rest, and then another 2 phrases. Each phrase is 4 bars long (although spread over 5 bars due to the *syncopation*) and they are rhythmically and melodically very similar.

Given the four bar structure of the phrases, a strong autocorrelation at lag $k = 4$ is expected. The autocorrelation should also identify the higher-level 32 bar (section + rest) structure. The longer rests between the fourth and fifth repetition of each phrase in the sections may cause some problems. The longer rest lasts two bars and so effectively places the last two phrases exactly out of sync with the preceding four of each section. By putting the last two phases out of sync with the first four, the autocorrelation process may also detect a two-bar structure when the first four bars are put out of phase with each other but in phase with the last two.

The curve-fitting process should, as before, identify the same features as the au-

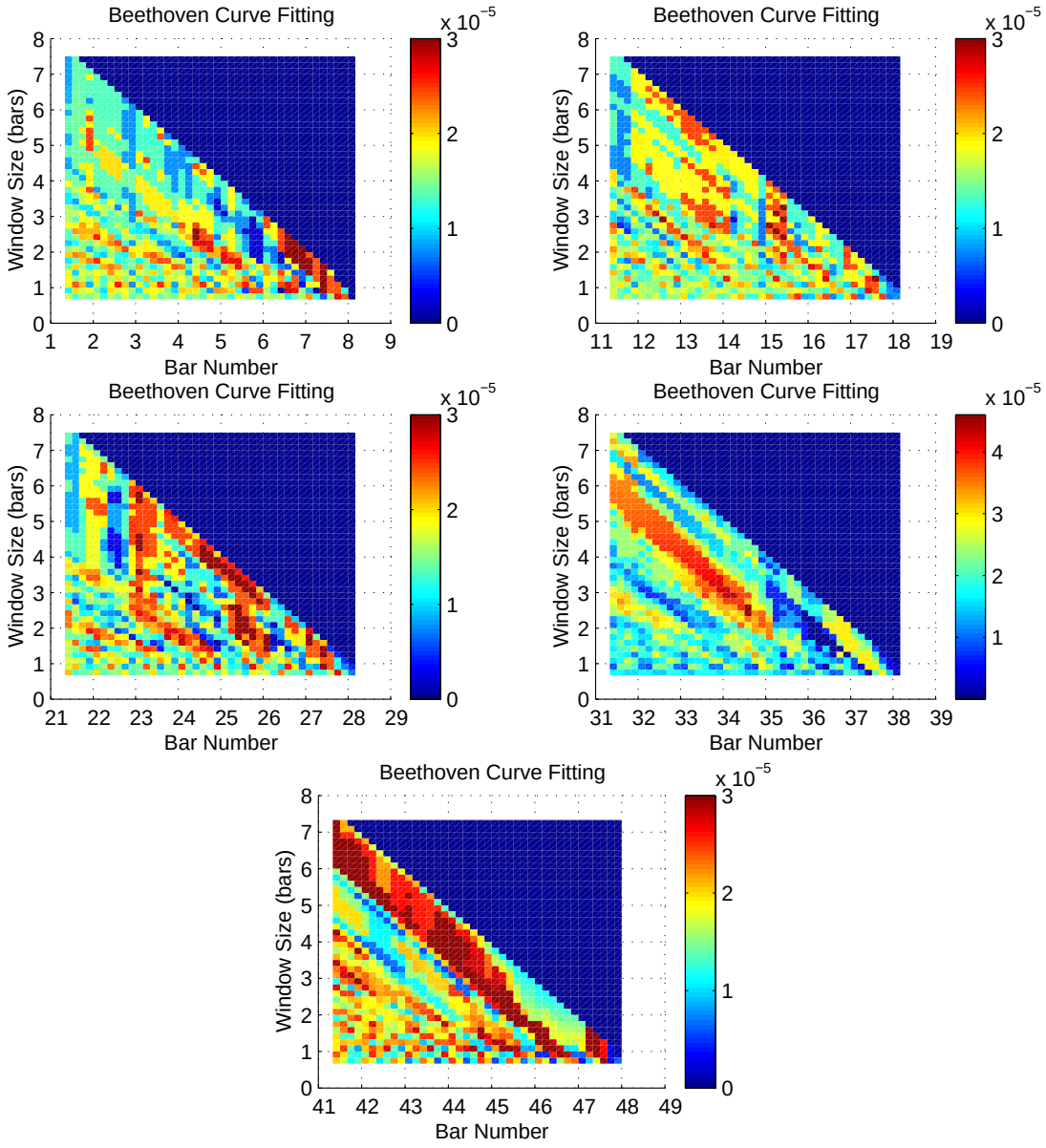


Figure 7.23: Detail of curve fitting results for sections of *Auf dem Hügel sitz ich spähend*.

tocorrelation process. There should be a horizontal pattern of strong and weak errors at around $y = 4$; however, due to the change in phase in each section the pattern will not be continuous but will alter according to the location within each section. There should also be features which identify the phrase structures.

7.4.3.2 Autocorrelation Results

Figure 7.24 shows the results of applying autocorrelation techniques to the performances of *Gute Nacht*. There is a very strong correlation at $k = 4$ plus other, weaker ones at $k = 32$ and $k = 36$. There are numerous smaller peaks. The partial autocorrelation retains the peak at $k = 4$ but the larger peaks at $k = 32$ and $k = 36$ have been significantly reduced. The partial autocorrelation has also increased the peaks at $k = 11$ and $k = 13$ which corresponds to the length of the last phrases of the sections plus the two-bar rest before them.

Figure 7.25 presents the results of the autocorrelation processes when applied to each of the constituent sections in turn. The autocorrelation of the first section provides strong results for lags $k = 4$ and $k = 8$. The partial autocorrelation removes the lag at $k = 8$ but leaves the strong lag at $k = 4$ and introduces another peak at $k = 10$.

The autocorrelation results of the second section again provide a very strong correlation at $k = 4$ and another strong correlation at $k = 2$. This lag at $k = 2$ is removed when partial autocorrelation is applied.

Finally, the last section provides strong autocorrelation values at $k = 2, 4, 6, 8$ and 10. The peak at $k = 4$ is not as pronounced as in the previous two sections. When the partial autocorrelation is measured, the peak at $k = 4$ remains, although again less strong, and there remain peaks at $k = 2, 6, 8$ and 10.

7.4.3.3 Curve-Fitting Results

Figure 7.26 shows the result of applying the curve-fitting process to the whole of *Gute Nacht*. The results show a similar mix of strong and weak results to those of the *Berceuse* analysis and in contrast to those of the *Auf dem Hügel sitz ich spähend* analysis. There are a number of large areas with low errors, around $(9, 28)$, $(41, 26)$ and $(69, 24)$. At $y = 2$ and $y = 5$ there are horizontal patterns of alternating weak/strong

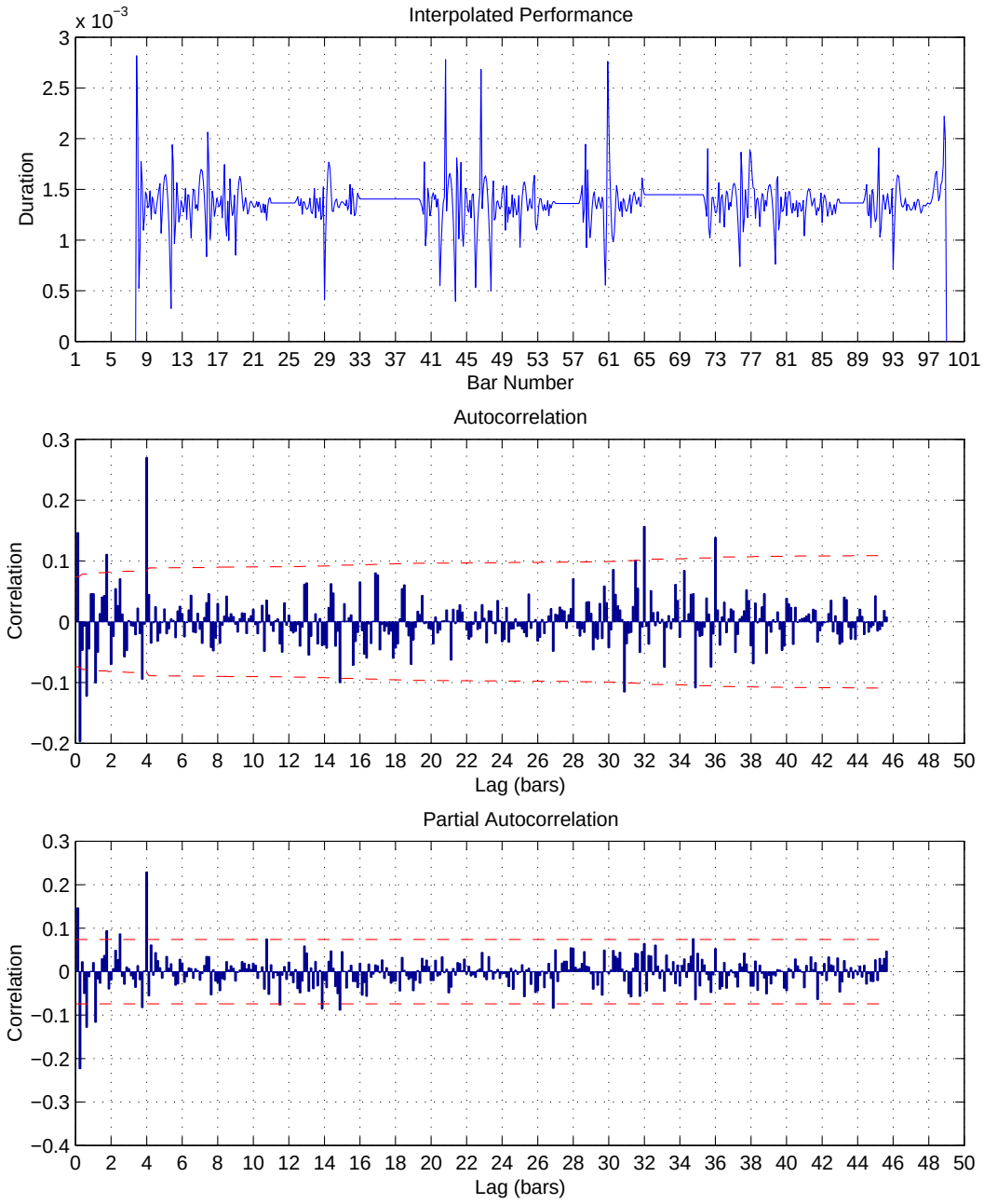


Figure 7.24: A graph showing the results of applying autocorrelation (middle) and partial autocorrelation (bottom) to the interpolated IOIs of *Gute Nacht*.

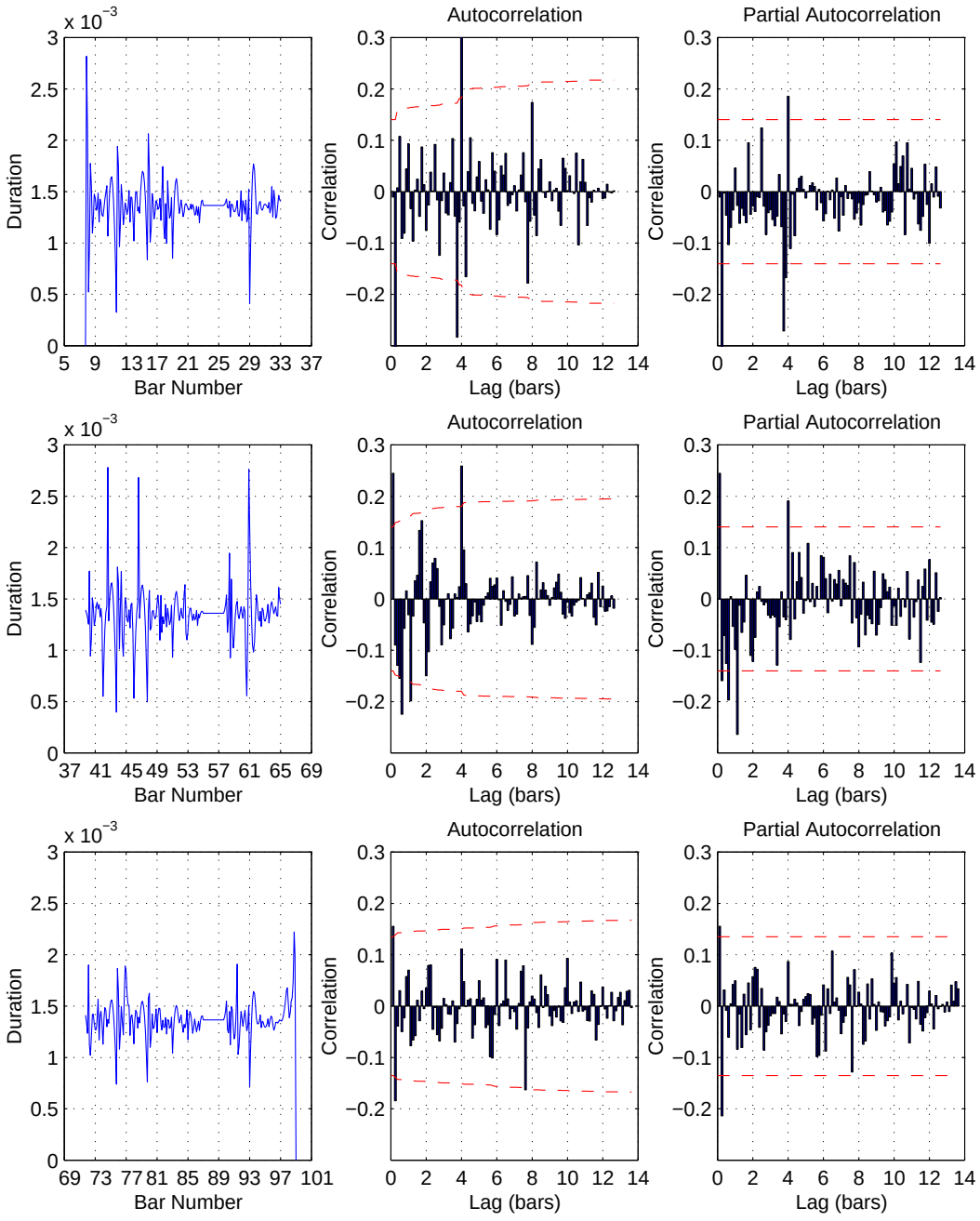


Figure 7.25: A graph showing the results of applying autocorrelation and partial autocorrelation to the interpolated IOIs of each section of *Gute Nacht*.

errors. As with the previous pieces, the same analysis process can be applied to the individual sections of the piece.

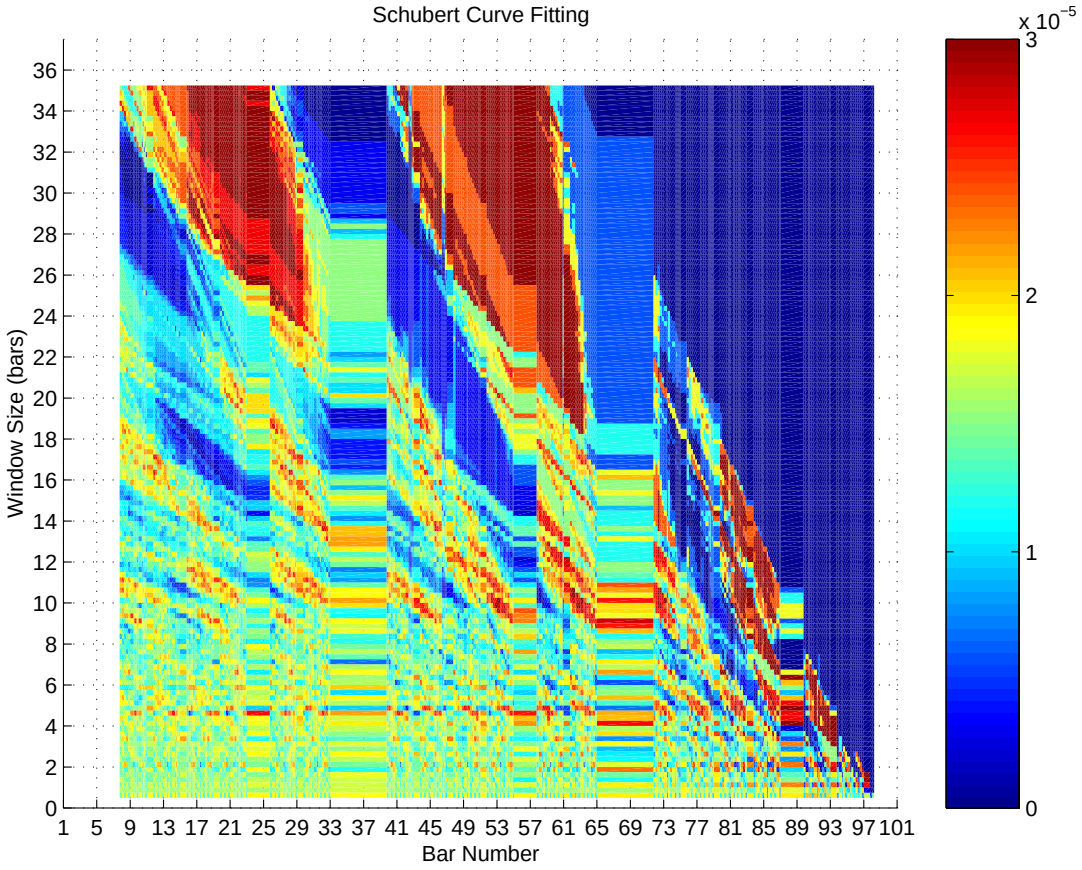
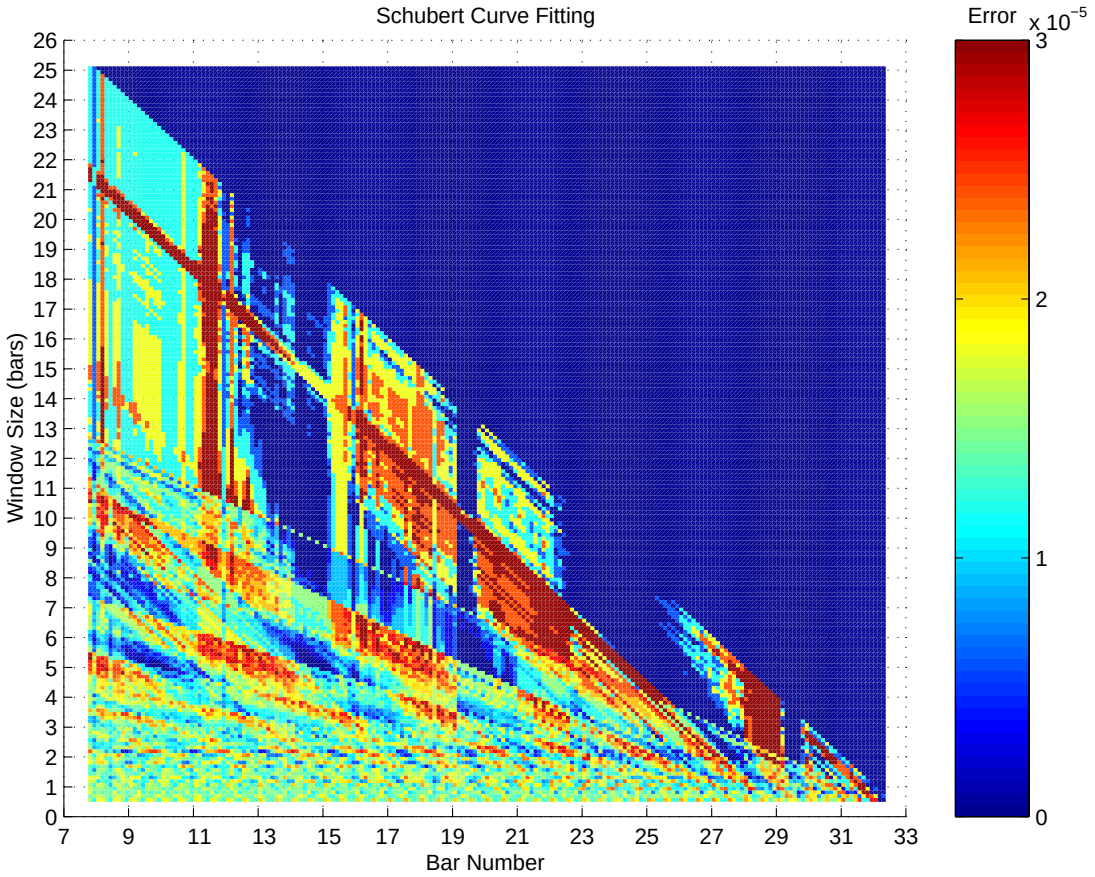


Figure 7.26: Curve fitting results for *Gute Nacht*.

Figure 7.27 shows the results of applying the curve-fitting process to the first section of the piece. Two large features dominate the results; one centred around $(14, 12)$ and the other at $(24, 7)$. Apart from these two large features, there are two distinct vertical patterns of low errors near $x = 19$ and $x = 29$. There is also evidence of the two-bar ($y = 2$) and five-bar ($y = 5$) patterns identified in the analysis of the whole piece. A distinct diagonal stripe of high errors cuts through the results from $(8, 21)$ to $(29, 0)$.

Figure 7.28 presents the curve-fitting results for the second section of the piece. Four vertical stripes of low errors occur near $x = 42,\ 46,\ 51$ and $x = 61$. There is



occurred

Figure 7.27: Detail of curve fitting results for bars 7–39 of *Gute Nacht*.

a weak horizontal pattern of alternating strength measurements at $y = 3$ and again at $y = 5$. Interestingly, the distinct diagonal stripe of high error values is replaced in this section by a similar stripe of low errors running from $(40, 21)$ to $(61, 0)$.

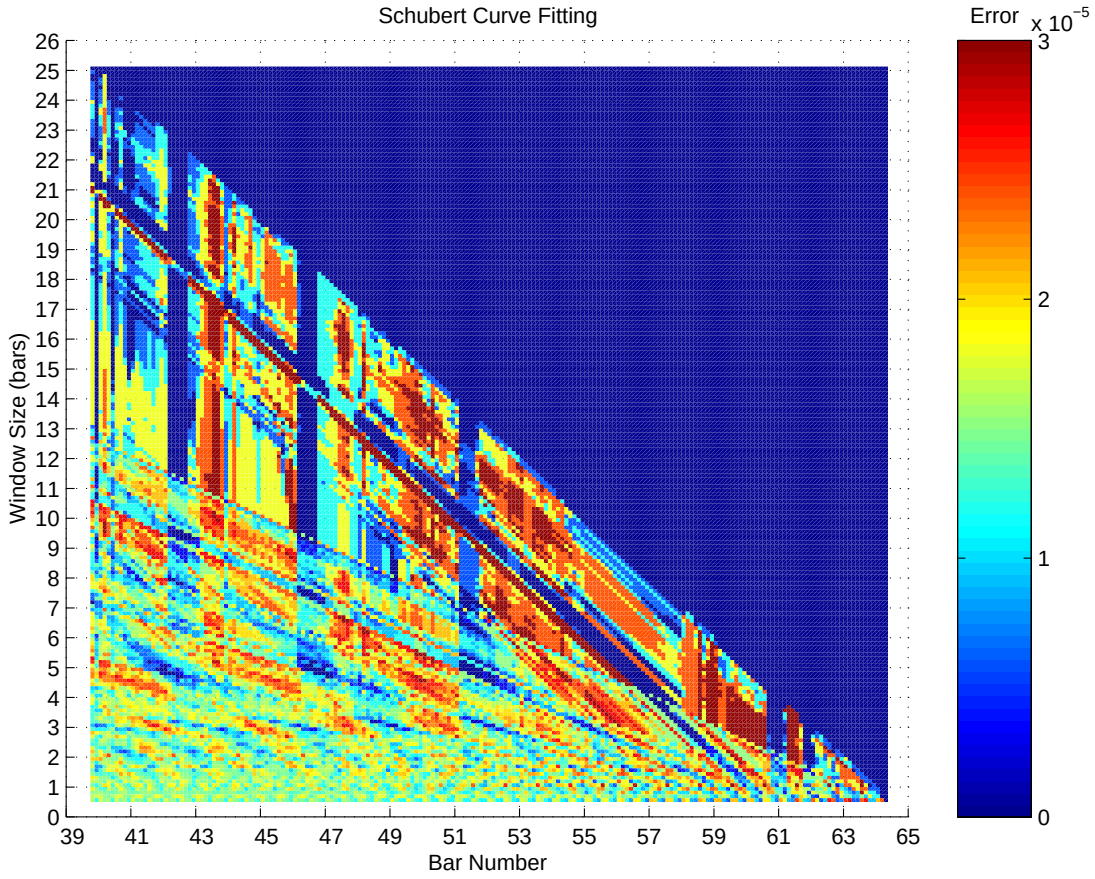


Figure 7.28: Detail of curve fitting results for bars 39–71 of *Gute Nacht*.

Finally, Figure 7.29 shows the results of the curve-fitting process when applied to the final third of the piece. There are three large scale areas of low errors based around $(74, 17)$, $(77, 14)$ and $(88, 7)$. Two vertical areas of low error occur near $x = 81$ and $x = 83$. The alternating pattern of weak/strong errors occurs at $y = 2$, however the accompanying pattern at $y = 5$ is less clear than for the previous results. Finally, the diagonal stripe of high error values has returned (spanning from $(72, 21)$ to $(93, 0)$).

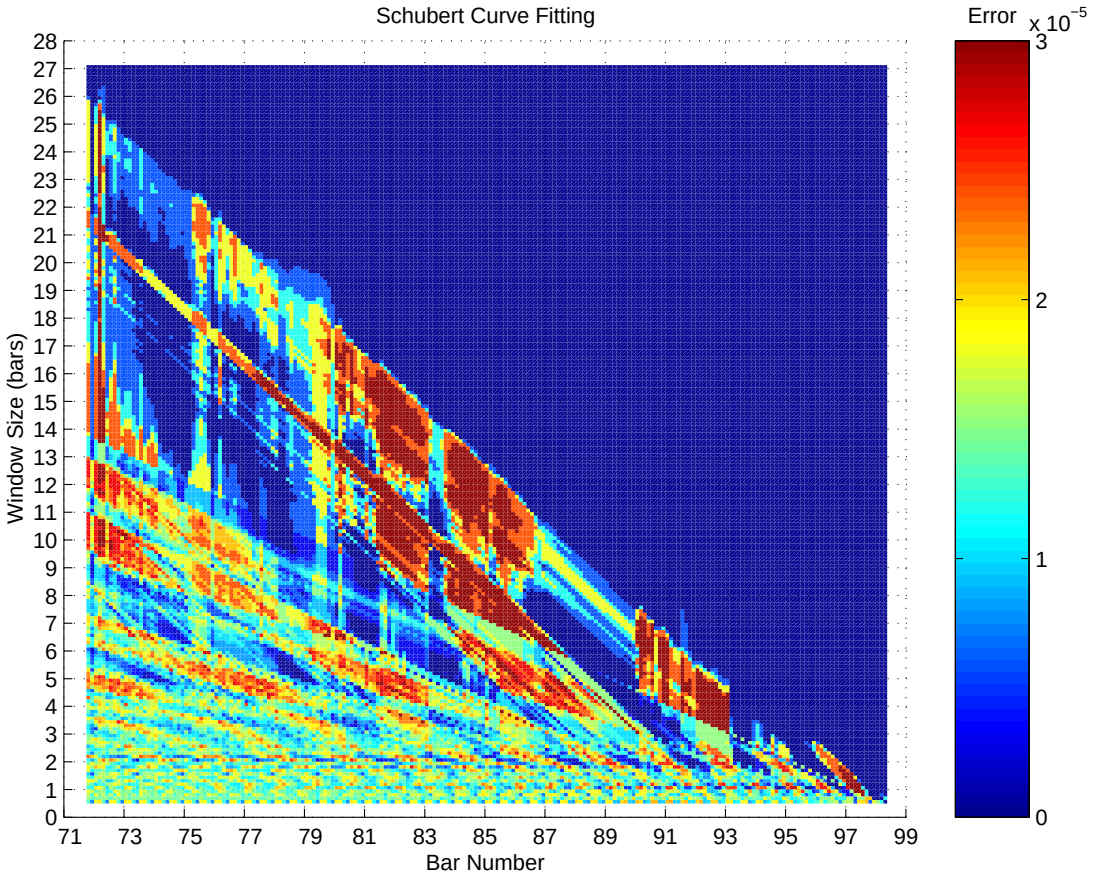


Figure 7.29: Detail of curve fitting results for bars 71–97 of *Gute Nacht*.

7.5 Summary and Discussion

In this chapter, the treatment of the recorded performance data was discussed including how the performance data was interpolated to provide a uniformly distributed set of data which was then analysed to identify structurally salient points of the performance.

The performance data was considered as a time series that represents the relationship between the duration of the performed events and how they are notated in the musical score.

The time series was then treated to provide a contiguous set of points which, as accurately as possible, reflected the original information recorded. This was done by applying two different methods of interpolation. First, a simple approach to interpolation was used which averages out the duration of an event over its notated length in the score. The second, novel approach used a curve based interpolation method to try to provide a more context sensitive approach to interpolation.

The results from the two different interpolation methods varied in success: where the simple interpolation resulted in stepwise curves or truncated peaks and troughs, the context sensitive interpolation produced smoother more continuous curves. However, the context sensitive method produced a number of artifacts that would not be musically sensible especially when interpolating over points with relatively little contextual information (i.e. rests).

The results from the analysis techniques were equally mixed. The autocorrelation techniques identified the repetitive structures which were sought. However, they over-generated and produced many possible correlations which were not appropriate given the score. The other problem with the autocorrelation techniques is that they did not give any positional information about where these structural units occur. For these performances, the information returned from the autocorrelation techniques is not sufficient to identify the phrasal structures of the pieces.

The curve-fitting analyses provided some very encouraging results but, in a similar way to the autocorrelation techniques, the results tended to have a lot of noise. However from an initial inspection of the results there do appear to be a number of interesting features which will be discussed in more detail in the next chapter.

Encouragingly, in terms of comparing the two techniques, when applying the two

techniques to the sets of performance data, the relative quality of the results returned seemed similar for each piece.

The next chapter presents how the results from both the autocorrelation techniques and the curve-fitting process can be incorporated into the rest of the system in order to disambiguate the possible grouping structures.

Chapter 8

Feature Detection

8.1 Introduction

The previous chapter presented a way of interpolating the raw performance data to produce an equispaced, discrete set of data. This data was then subjected to a curve-fitting analysis which specialised in detecting occurrences of convex curves. The curve-fitting analysis process was applied to the performance data and produced a graph that showed, for each pairing of starting position and window size, how closely the performance data could be approximated by a convex curve.

This chapter describes how these results from the curve-fitting process can be analysed in order to locate significant features which aid the identification of the musical structure. The significant features correspond to a series of three different patterns which may arise in the curve-fitting analysis. A process by which these patterns may be detected from the data is presented. A full implementation of this process has not been attempted as part of this research. However, one of the aims of this chapter is to provide sufficient detail to allow such an implementation to be achieved.

Figure 8.1 shows the rôle of the research presented in this chapter within the structural disambiguation process. The results of applying the process described in this chapter will be a set of features that can inform the decision about which grouping boundaries should be considered active.

The chapter begins with an overview of the kinds of features which can be identi-

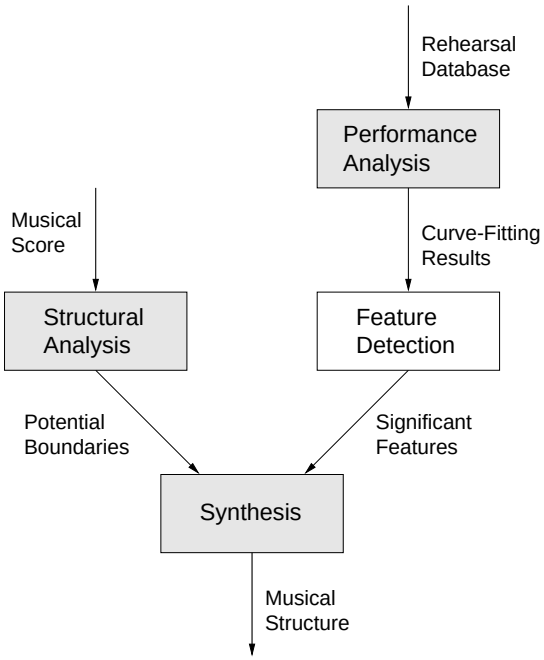


Figure 8.1: Pictorial representation of the rôle of this component within the structural disambiguation flow.

fied in the performance analysis.

8.2 Patterns

An individual point which has a low curve-fitting error is not necessarily indicative of an important structural feature in the performance. However, when a set of such points forms a well defined pattern, as described below, that set of points may indicate an interesting performance feature.

As mentioned in Section 7.4.1.3, there are a number of different patterns which can be searched for in the graphs of curve-fitting results:

- Vertical lines that, overall, tend to container stronger curve-fits;
- Diagonal stripes that, similarly to the vertical lines, tend to contain stronger curve-fits;
- Horizontal lines which contain alternating strong and weak curve-fits.[1]

These three patterns are the primary source for information which will be used to support or reject the possible structural boundaries identified by the structural analysis. The following sections describe each of these patterns and what they indicate with respect to the performance data.

8.2.1 Vertical Features

Figure 8.2 illustrates how a vertical pattern of low errors would arise from the curve-fitting data and how that relates to the original performance data. The top half of the figure shows a section of performance data for a piece with four and eight beat phrasal units. For example, from $x = 8$ to $x = 12$ constitutes one four-beat phrase, as do the events from $x = 12$ to $x = 16$. Similarly, there is an eight-beat phrase from $x = 8$ to $x = 16$ and from $x = 16$ to $x = 24$.

Now if a particular starting position is chosen, say $x = 8$ (as shown by the vertical dashed line in the top of the figure), there are a number of sizes of convex quadratic

[1] Note that *strong* in this situation is equivalent to the curve fitting process returning a small error.

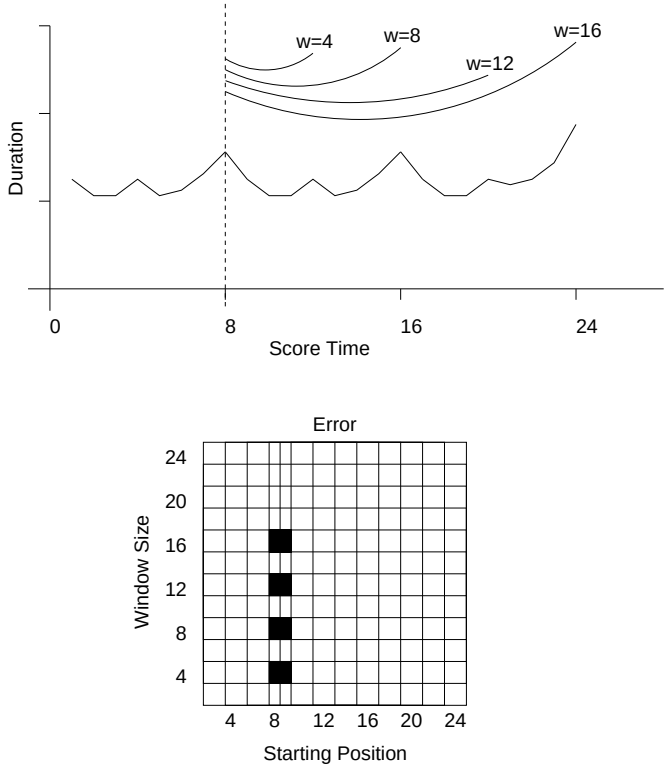


Figure 8.2: Illustration of how a number of good curve-fits which start from the same point in the performance lead to a vertical feature in the performance analysis.

curves that provide a good approximation for the subsequent events. In this case, window sizes of $w = 4, 8, 12, 16$ all provide curves that closely approximate the curve underlying the performance data of those following events.

Combining this starting value with these window sizes implies that at points $(8,4)$, $(8,8)$, $(8,12)$ and $(8,16)$ in the curve-fit analysis there should be a low error. These low errors will manifest themselves as a vertical column of blue points in the curve fit results or, as will be seen in Section 8.3.1, a vertical column of dark points in the thresholded results.

8.2.2 Diagonal Features

Another feature, similar in nature to the vertical features, is diagonal patterns of low errors in the curve-fitting results. Figure 8.3 illustrates how these would occur in the curve-fitting results and how they arise from the performance data.

A diagonal feature corresponds to a shared ending point for a set of curves. From the figure, the point at $x = 24$ is a common ending point for the four curves of size $w = 4, 8, 12$ and $w = 16$. Each of these curves has a different starting point, namely $x = 20, 16, 12$ and $x = 8$ respectively. The combination of these x and w values lead to a diagonal stripe of dark points at $(20,4)$, $(16,8)$, $(12,12)$ and $(8,16)$ (as shown in the bottom half of the figure).

Such a set of points can be expressed more generically as Equation 8.1 where x is the starting point, w is the window size and $\eta_{x,w}$ is the measured error for that starting point and window size. Note that the sum of the x and w values (equivalent to the abscissa and the ordinate of the coordinate form) equal a constant value k.

$$S = \{\eta_{x,w} \mid x + w = k\} \tag{8.1}$$

This value of k represents a place in the score which all these windows share and which corresponds to the common ending point of the set of windows contained in S (i.e. when $w = 0$ then $x = k$). Because this set of windows is formed by points which have a low error, it suggests that k must be the shared ending of the curves and is a potentially interesting feature.

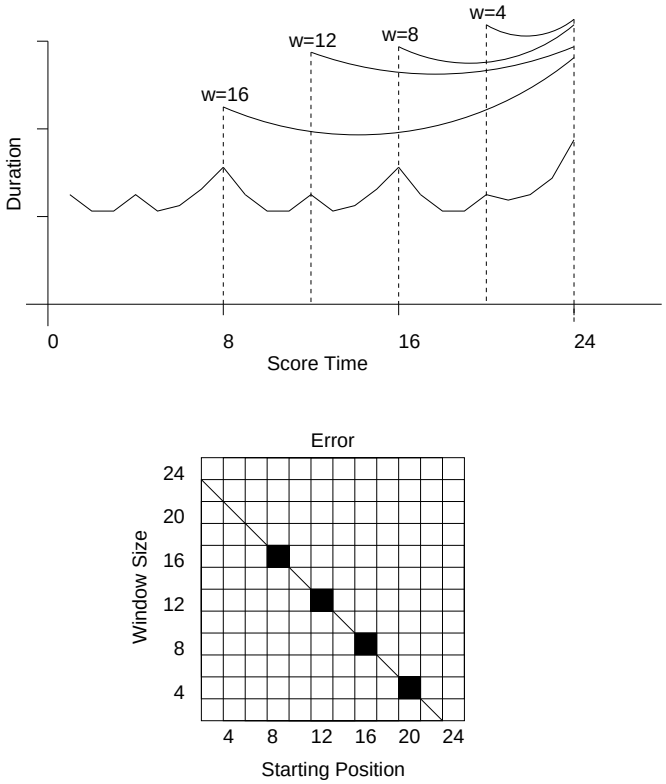


Figure 8.3: Illustration of how a number of curves which end at a shared point lead to a diagonal feature.

8.2.3 Horizontal Features

The final feature consists of a horizontal line of errors which alternate in strength from low to high. This feature would arise from a particular window size going in and out of phase as it was applied across the data. Figure 8.4 illustrates how this may arise from a set of performance data.

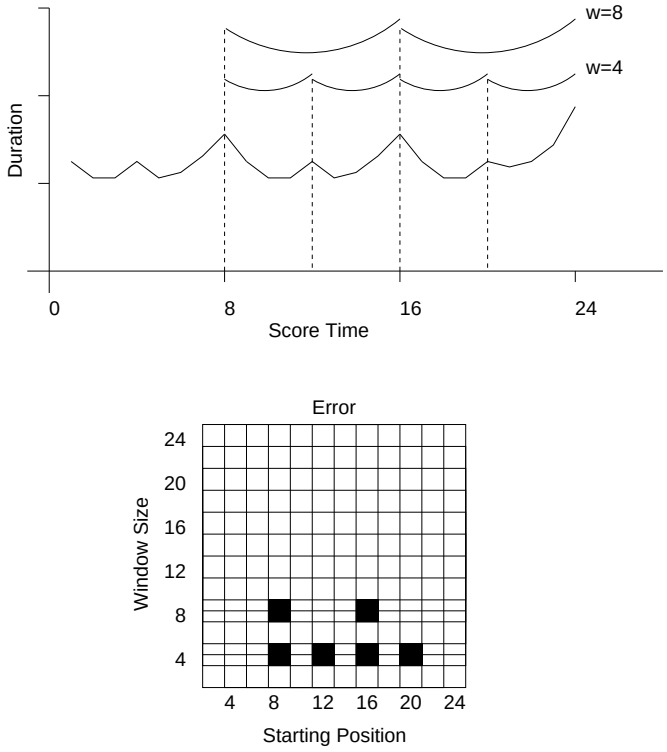


Figure 8.4: Illustration of how a repeating pattern of curves that goes in and out of phase result in a horizontal feature in the performance analysis.

The figure shows how the performance data can be approximated by two curves of different lengths. The curve of length $w = 4$ can closely match the data when it is placed at $x = 8, 12, 16$ and $x = 20$. Similarly, the curve of length $w = 8$ closely models the data when it is placed at $x = 8$ and $x = 16$. The result of how these curve-fits would present in the curve-fitting results is shown in the bottom half of the figure. They appear as two lines of points parallel to the x-axis at $w = 4$ and $w = 8$.[2]

[2]Incidentally, examples of the previous two features can be seen as vertical columns at $x = 8$ and

The next sections describe how these features can be identified in the results from the curve-fitting analysis of the three pieces described in Chapter 7.

8.2.4 Feature Detection in Practice

The examples in the previous sections illustrated the three features arising in ideal situations - the features are clearly present and no noise was present in the results. This is not true in general.

There are three types of pattern which are of interest: vertical or diagonal lines of low error and horizontal patterns of alternating error strengths. The first step in identifying these patterns is to threshold the data from the initial curve analysis to clearly identify those areas of the results that have low errors. The resulting binary data is then examined to identify occurrences of the aforementioned three patterns.

The results from the analysis process are not expected to produce perfect results and the identification of features remains a manual process in this research. The following criteria were used to identify patterns in the thresholded data:

- A sharp vertical or $-45°$ edge of a large solid mass;
- A series of islands which are aligned either vertically or at $-45°$;
- A combination of the above two patterns, such as a long edge with a number of islands which lie on the line formed by that edge;
- A series of islands horizontally aligned and separated by an amount approximately equal to their height on the y-axis.

There are a number of important terms, e.g. ‘sharp’ or ‘large’, in the above criteria which would need to be more concretely defined before a fully automated process could be used to identify the patterns. For the manual process used here, an edge was generally considered ‘sharp’ if it only deviated from a pure vertical or diagonal line by a quarter of a bar in either direction. Similarly, for this research, ‘large’ corresponded to those features that occupied approximately a third or more of the available space.

$x = 16$, and as diagonal stripes ending at $x = 16$ and $x = 24$.

Where the available space will depend on the x position for the vertical features and the x and y intercepts for the diagonal features.

Vertical and diagonal features share a common set of requirements. As described above, these two features should consist of either a sharp edge, a series of islands or some combination of the two. For these features, the identification had to take into account the local context. For example, a feature that spanned two bars on the vertical axis would not probably be considered significant for a low x value whereas for a high x value, where the potential size of any feature is limited, it would be considered important (e.g.V8 in Figure 8.6).

A typical horizontal feature consists of a number of islands which lie at the same vertical height and which are separated by a fixed amount. A strong preference is given to a chain of islands whose horizontal separation precisely corresponds to their height on the y-axis. For example, in Figure 8.7 the horizontal feature at H6 was chosen as there is a chain of islands at $y = 2$ which are horizontally separated by two bars.

If the total length of the excerpt under analysis is notated by l, it is impossible to identify a chain of islands which are separated by any distance greater than $l/2$ as that precludes the chance of more than one island occuring within the scope of the data. For example, in Figure 8.7, no chain of islands spaced more than 12 bars apart could be identified. As a general rule, only series of three or more islands were identified as valid horizontal features. This meant that horizontal features were only searched for in the lower third of the results.

8.3 Results: *Berceuse*

This section describes how the results from the curve-fitting analysis of *Berceuse* were analysed to detect significant patterns. The section begins by describing the use of thresholding to aid the detection process.

8.3.1 Thresholding

In order to simplify the feature detection process, a threshold is applied across the curve-fitting results in order to select only those parts of the performance analysis that

have low-errors. To select an appropriate threshold, a frequency graph was plotted for each of the piece's sections. Figure 8.5 shows the number of occurrences of each value of measured error in the performance analysis.

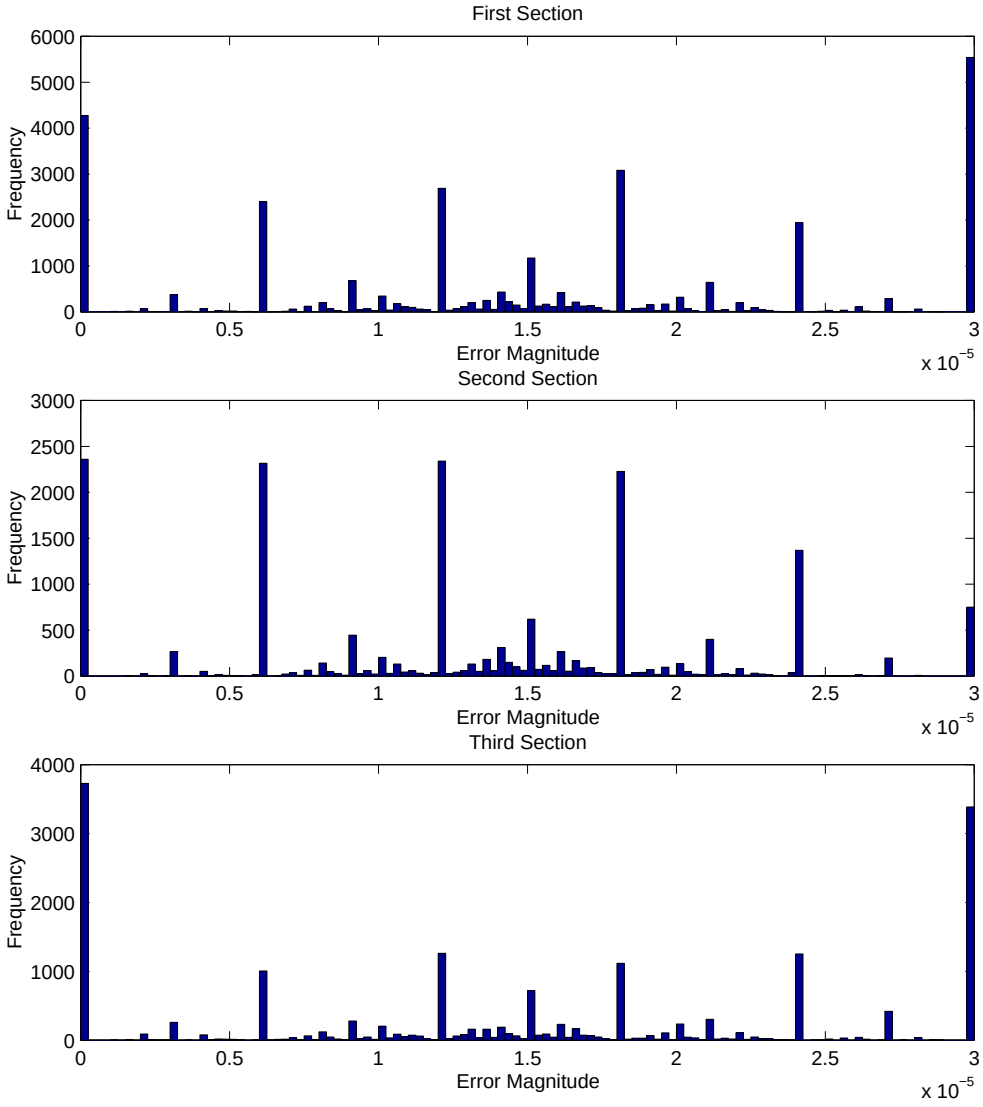


Figure 8.5: Frequency distributions for the curve-fitting results of *Berceuse*.

The unusual shape of the distribution, i.e. the peaks at $x = 0.6, 1.2, 1.8, 2.4 \times 10^{-5}$, is due to cases in the performance data when, for some of the performances, the quadratic modelling provided a good fit and for the other performances a very poor

fit. For example, if in one performance, the curve-fitting process identified a good convex curve and in the other four performances a concave curve was discovered, then the average error for that location would be dominated by the error assigned to concave curves (see Section 7.3.2).

For example, during the analysis of *Berceuse*, the fixed error assigned to identified concave curves was chosen to be 3×10^{-5}. The peak at $x = 0.6 \times 10^{-5}$ originates from the average error occurring when the performance-analysis process identified four concave curves ($\eta = 0.0$) and one convex curve ($\eta = 3 \times 10^{-5}$) which results in an average error of $x = 0.6 \times 10^{-5}$. The other four peaks mentioned above can be explained in a similar manner.

The chosen threshold should eliminate from the graph those points of relatively high error which do not contain structural information which can be used during the disambiguation process.

For this research, a threshold of 0.7×10^{-5} was chosen which would select those combinations of window-size and starting position that produced a low error for at least four of the five performances. The thresholded results contain the features that were visible in the original performance analysis results and have reduced the amount of extraneous information on the graphs.

8.3.2 Features in Section One

The result of applying the threshold to the curve-fitting data of the first section of *Berceuse* is shown in Figure 8.6. The black points represent an average curve-fitting error that was below the threshold and is therefore considered significant as it represents a point of low error which occurred in at least four of the five performances.

The resulting graph contains three distinct areas. The first area, from bar 3 to bar 10 contains a large island of low curve fitting errors. The second area, from bar 10 to bar 18, contains relatively few points of low errors. Finally the last section of the graph, from bar 18 to bar 34 consists of patterns of islands that have formed along diagonal lines.

Occurrences of the three patterns of results which are of interest have been marked on Figure 8.6. The following sections discuss the marked occurrences of each of these

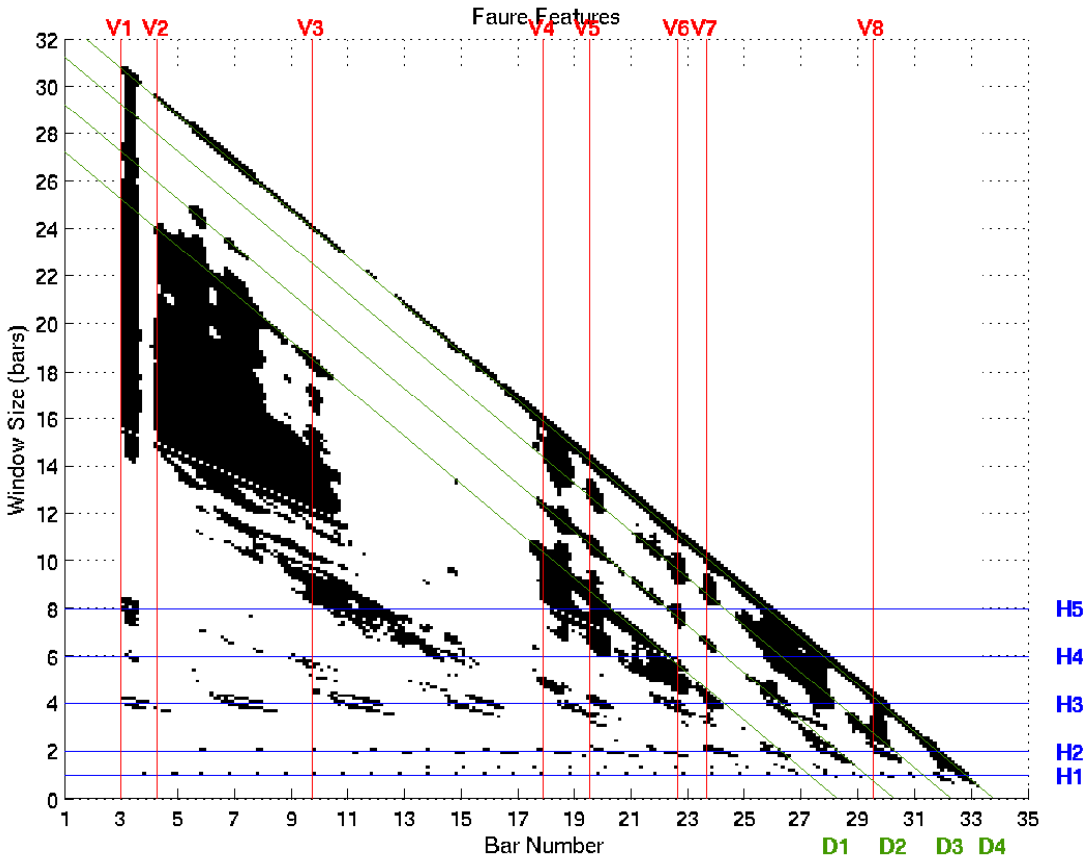


Figure 8.6: Thresholded curve-fitting data with feature annotations for the first third (up to bar 34) of *Berceuse*.

types of patterns.

Vertical Patterns

There are eight vertical patterns of low errors apparent in the figure. The first of these is the large vertical stripe (V1) at $x = 3$ at the very beginning of the piece which arises from the sharp edge of the vertical island of low errors spanning from $(3, 14)$ to $(3, 31)$. The second (V2) occurs just after the start of bar 4 at the distinct vertical edge of the large expanse of low errors based around $(7, 16)$.

Near $x = 6$, there is a hint of a vertical edge or chain of islands at $(6, 20)$ and $(6, 22)$ however the evidence for this is comparitively weak when compared with the surrounding features. So this potential vertical feature remains unselected.

The third vertical pattern (V3), occurring just before bar 10, is selected due to the three islands of low error at $(10, 16)$, $(10, 17)$ and $(10, 19)$, and the intersection with the edge of the mass at $(10, 9)$.

Another potential feature could be considered at $x = 15$ due to the small islands at $(15, 14)$, $(15, 10)$ and $(15, 7)$. However, considering the size of these features which would be represented by the first two of the above islands (i.e. 14 and 10 bars wide respectively) they appear as very small islands when compared with islands of a similar height in the rest of the analysis. For this reason, they are considered to be false positives.

The feature at $x = 18$, labelled as V4 is chosen by the combination of the sharp edge of the island at $(18, 10)$ and the intersection with the edges of two islands above at $(18, 13)$ and $(18, 15)$, and the island beneath at $(18, 5)$.

Another strong candidate feature could be considered at $x = 18.5$ where the islands at $(18.5, 10)$, $(18.5, 12)$ and $(18.5, 14)$ suggest the possibility of a feature. However, given the clearer features either side, in close proximity, and the relative lack of definition of where exactly the candidate edge would lie, this feature was omitted.

The next feature (V5), at $x = 19.5$ occurs due to the islands of low-errors at $y = 4, 5, 8, 11$ and $y = 13$. There are two other similar features (V6 and V7) created either side of $x = 23$ and finally, the eighth feature (V8) occurs on the edge at $x = 29.5$. This last feature, V8 has relatively weak evidence for its inclusion and would not have been

selected if it had occurred at a lesser x value, however due to its relative strength in such a limited space, it was selected.

Diagonal Patterns

There are four diagonal features, marked D1 to D4 which are present towards the later part of the graph. The most pronounced of these features is D4 (terminating at bar 34) which marks the end of this section of the piece. The feature marked D1 serves as a boundary to the large island of low errors (near $(7,16)$) and to another island of errors at $(20,8)$.

The remaining two features D2 and D3 present as two diagonal stripes of islands which occur predominantly in the second half of this section. There are no other strong diagonal candidates occurring at $-45°$.

Horizontal Patterns

Five diagonal features are present in the thresholded results (H1 to H5). Four of these features are particularly pronounced, occurring at $y = 1,2,4$ and $y = 8$. For example, at $y = 1$ there are a series of islands beginning at $x = 4$ and repeatedly occurring at every bar until $x = 33$. The remaining feature, at $y = 6$, is less pronounced and consistent, and is probably due to the interaction of the lower level features (in a similar way to the autocorrelation process).

8.3.3 Features in Section Two

Figure 8.7 shows the results of applying a threshold to the curve-fitting results of the middle section of *Berceuse*. The most obvious features are the vertical line of low errors at the very start of this section (bars 35–36) and the diagonal stripe of low errors running from $(35,23)$ to $(58,1)$. Near the centre of the results are two islands of results centred around $(42,13)$ and $(44,10)$.

As before, the interesting features have been marked on the figure and are discussed below.

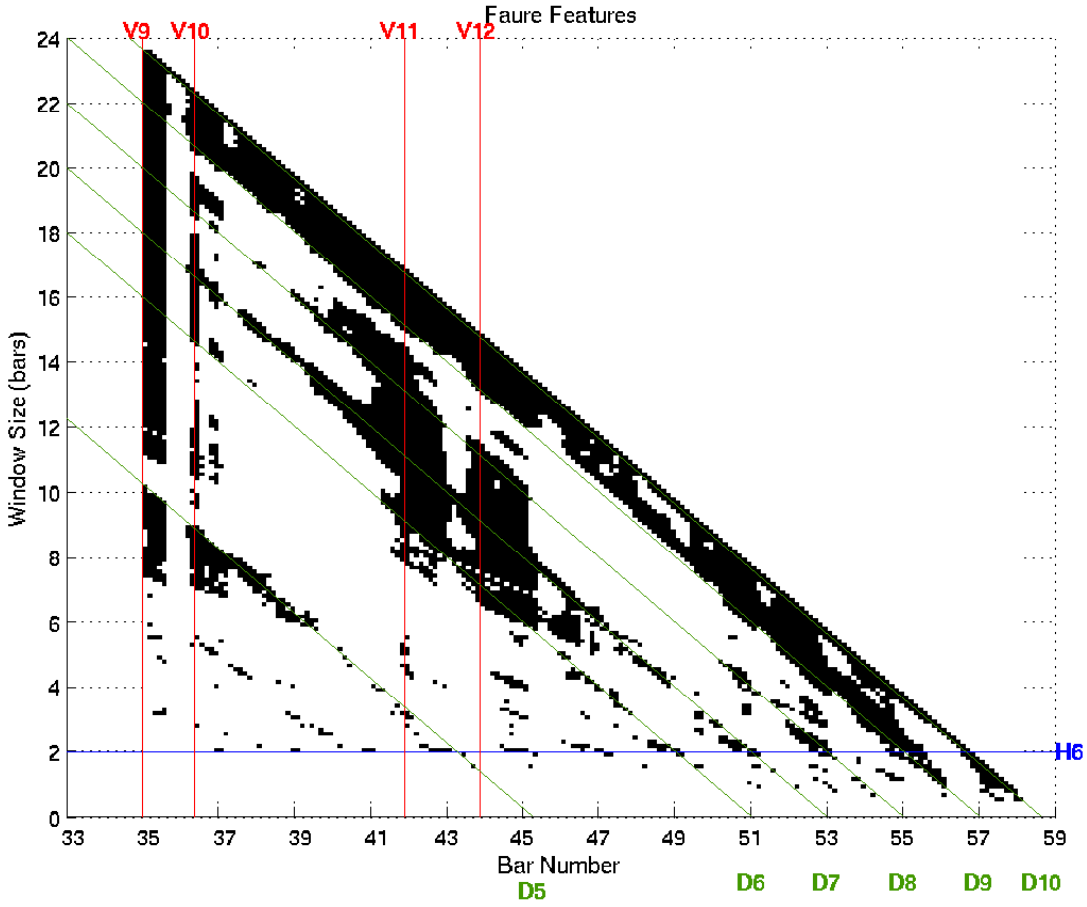


Figure 8.7: Thresholded curve-fitting results with feature annotations for the middle section (bars 35–59) of *Berceuse*.

Vertical Patterns

There are two clear vertical features (V9 and V10) at the start of the section at bars 35 and 36. The first of these features is clearly indicated by the two large islands of points with edges at $x = 35$. The second feature, V10, is formed by a long chain of islands with edges lying at $x = 36$.

At $x = 37$ these is another strong suggestion of a vertical feature which could possibly be included but is not due to it's close proximity to V10 and its relatively undefined edge.

In the centre of the results there are two more features implied by the edges of the two islands of low errors. These edges marked as V11 and V12 occur at bars 42 and 44. Although these two large masses have relatively undefined edges, they are distinctly present as vertically orientated islands. Due to the irregularity of their edges, it is difficult to identify a clear starting point for the vertical feature and the features are chosen to pass through the leftmost part of the islands which have the most low errors and in the case of V11 to pass through the islands at $(42,4)$ and $(42,5)$. There are no other vertical features.

Diagonal Patterns

The first diagonal pattern (D5) runs from $(35,10)$ down to $(45,0)$ and is highlighted by the edge of the island at $(37,8)$. Again, the line is chosen to pass through as many points of low error as possible.

There are then five diagonal patterns (D6–D10) terminating at bars 51, 53, 55, 57 and 59. Three of these patterns are particularly clear (D7, D9 and D10) and are formed by stripes of low-errors that extend through most combinations of starting point and window-size.

The other two patterns are formed though the alignment of a number of smaller islands. D6 is formed by the edges of the islands near $(37,14)$ and then passing through the bottom edges of the two large masses in the centre of the figure and finally running though the two diagonal clusters of islands at $(47,4)$ and $(49,2)$. The diagonal feature D8 arises from the top edges of the two large masses and then intersecting two small diagonal groups of islands at $(51,5)$ and $(53,2)$.

Horizontal Patterns

There is only one clear horizontal (H6) pattern occurring at $y = 2$ formed by the chain of islands beginning at $x = 37$ and repeating every two bars until bar 57..

8.3.4 Features in Section Three

The three main features that stand out from the thresholded results for the third section of *Berceuse* (shown in Figure 8.8) are a long diagonal stripe running from $(59, 24)$ to $(82, 1)$, a pair of vertical features near bar 73 and a large diagonal island centred around $(62, 14)$.

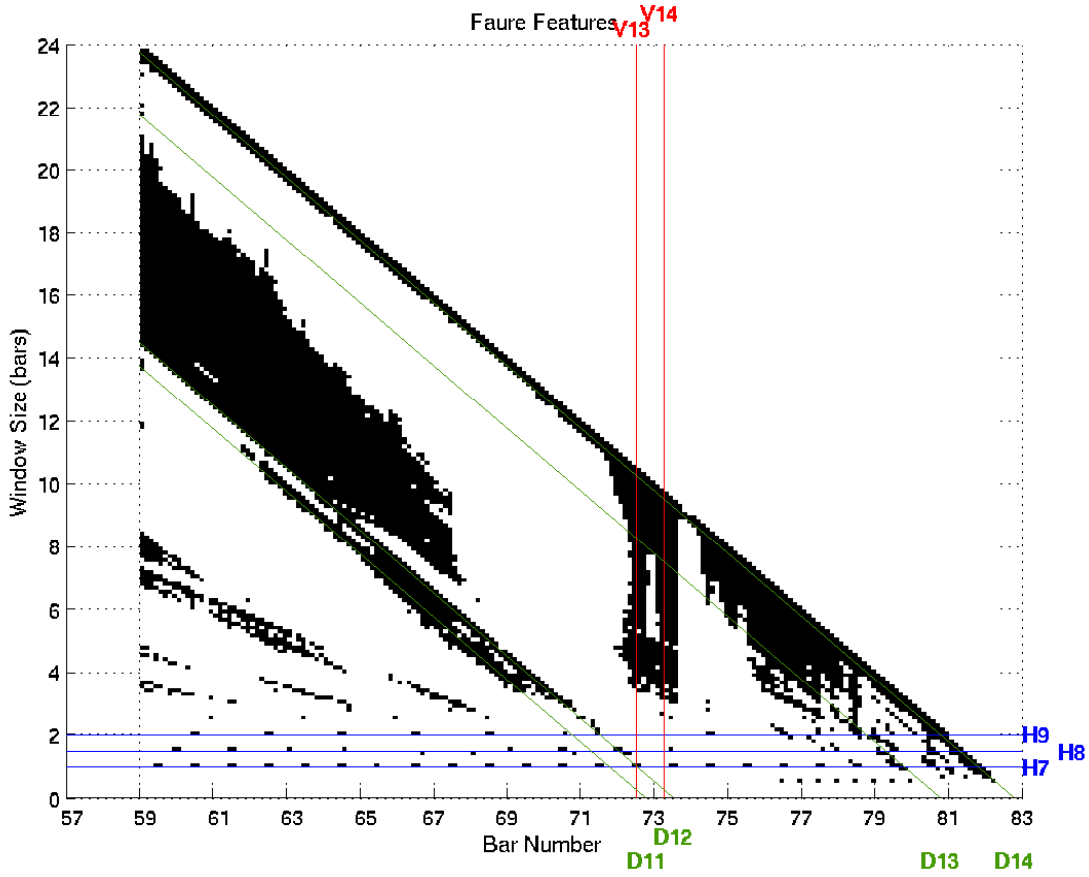


Figure 8.8: The annotated and thresholded curve-fitting results for the final third (bars 59–83) of *Berceuse*.

The following sections describe the occurrences of the patterns that have been marked on the graph.

Vertical Patterns

There are only two vertical patterns which occur for this section. The first of these, V13, occurs just after the start of bar 72. The second (V14) occurs in bar 73 approximately one bar after the previous feature. These two features are formed by the two vertical islands at $(72,6)$ and $(73,6)$.

Diagonal Patterns

There are four diagonal patterns marked on the figure. Three of these (D11, D12 and D14) are particularly strong. The first pair (D11 and D12) of patterns occur very close to each other and are created by the large diagonally oriented island (at $(63,12)$) with the extended diagonal edge and the parallel feature below it. The third strong feature, D14, extends across the entire set of bar and window size values and leads towards the end of the piece.

The weaker diagonal feature (D13) is created by the interaction of the islands in the last ten bars of the piece which imply the existence of such a feature. The feature does not have supporting clusters of points in the first fourteen bars of this section.

There are two other diagonal features which deserve discussion. The first of these is formed by the top edge of the large island at $(63,12)$. This topmost edge is so badly fragmented that there is no defining line which could be placed.

The second feature arises from the islands spanning from $(59,7)$ to $(68,3)$. Although there is a strong indication of a line of islands, these islands do not lie at the required -45° angle and so are not included as a feature.

Horizontal Patterns

There are three horizontal patterns of alternating weak/strong errors. The first of these, H7, occurs at the one bar window size ($y = 1$) and occurs strongly throughout the section. The second pattern, H8, occurs at $y = 1.5$ and is not as consistent as the first; it is only present as small continuous sections which are out of step with each other.

The final horizontal feature at $y = 2$, labelled H9, occurs strongly through most of the piece, and coincides with the finest grained feature (H7).

8.4 Results: *Auf dem Hügel sitz ich spähend*

The pattern identification process was applied to the performance data of *Auf dem Hügel sitz ich spähend*. The following sections describe the results beginning with a description of the thresholding process as applied to this piece.

8.4.1 Thresholding

Figure 8.9 shows the frequency distributions of the recorded errors from the curve-fitting analysis of *Auf dem Hügel sitz ich spähend*. The piece has been divided into five sections corresponding to the five stanzas of the song.

The resulting error distributions are markedly different from the distributions corresponding to *Berceuse*. The first of these differences relates to the overall shape of the distributions. The *Berceuse* results were dominated by the six peaks corresponding to the average errors caused by possible combinations of good/bad fits across the five pieces. The results for *Auf dem Hügel sitz ich spähend* are more evenly balanced with the peaks appearing less dominant with respect to the surrounding values.

Another interesting feature occurs in the fourth section of the piece where the error distribution extends beyond the 3×10^{-5} boundary to almost 4.5×10^{-5}. For all the other sections and for all the other pieces, the maximum error obtained by the curve-fitting process was 3×10^{-5}. The large error in this section of the piece is due to the interpolated section in bar 35 which has been interpolated towards zero and varies considerably from the average performance. The measured error between the convex curve which provides the best fit and the actual data is large enough to break the 3×10^{-5} limit.

The same thresholding limit of 0.7×10^{-5} was applied to the final data in order to select those strong curve-fits which were common to at least four of the performances. The features that could be identified from the thresholded data are described in the next section.

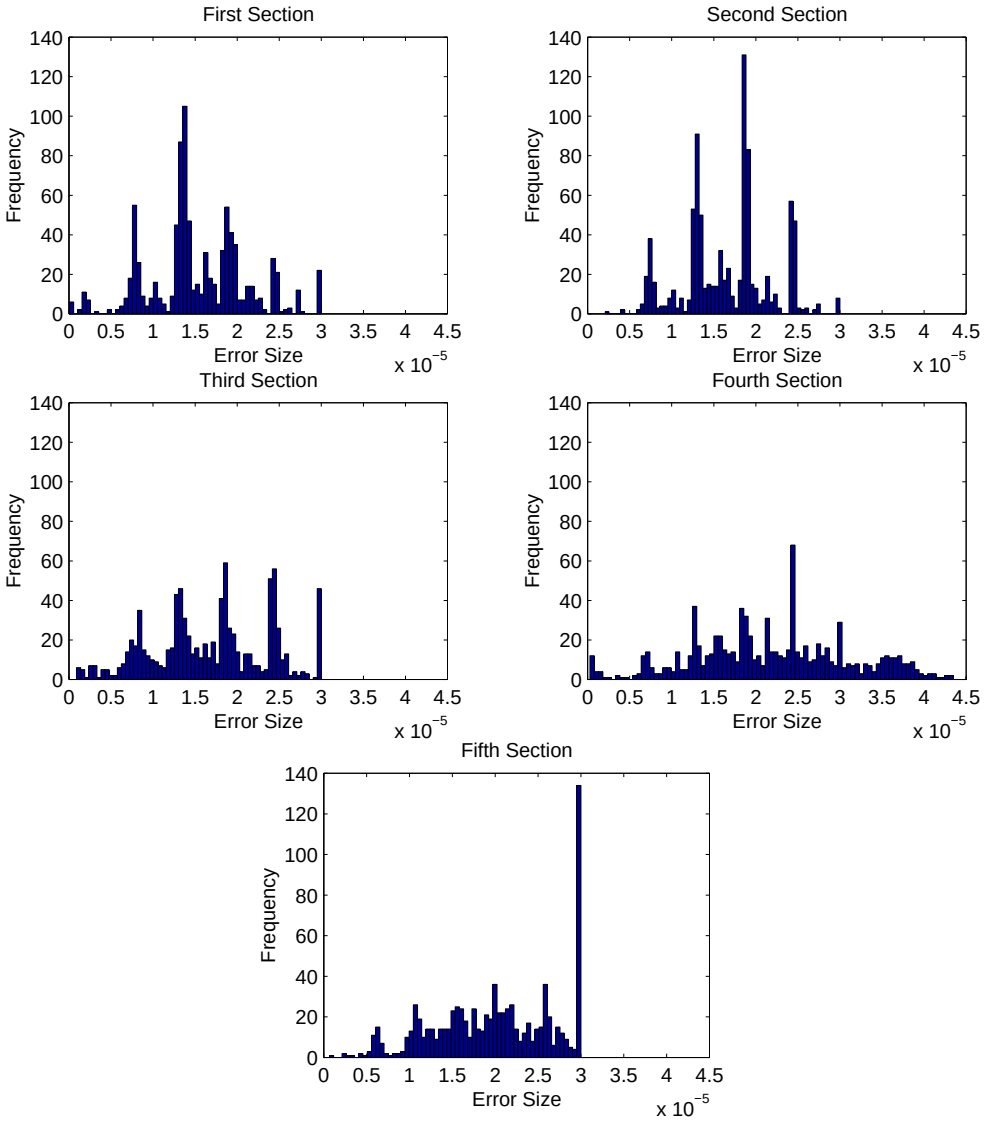


Figure 8.9: Frequency distributions for the curve-fitting results of *Auf dem Hügel sitz ich spähend*.

8.4.2 Features

Figure 8.10 shows the resulting thresholded performance analysis data along with the identified features. It is clear that the amount of data which is available for the feature detection is considerably less than for the other pieces and so the requirements of the feature detection process have been relaxed.

The following sections describe the features which are present in the thresholded results. Due to the small number of features identified, the piece will be discussed as a whole rather than as individual sections.

Vertical Patterns

There are four vertical features which all occur in the first three sections of the piece. The first feature (V1) occurs immediately before bar 6 and is caused by a tall island of points near $(2, 6)$. The second feature, labelled (V2), occurs on the border between bars 14 and 15, and is caused by another tall island of points.

The third and fourth vertical features both occur in the third section of the piece. The first of these, V3, is caused by both a tall island and above it another island which is vertically aligned with the tall island. This feature intersects the horizontal axis halfway through bar 22. The final feature, V4, is caused by a cluster of five islands which form a vertical column on the border between bars 24 and 25.

Diagonal Patterns

Ten diagonal features can be found in the results from the thresholded performance analysis. There are two features in the first section; the first (D1) is created by four aligned islands and is moderately strong, the second (D2) is much weaker and is only implied by a pair of islands. In the second section of the piece, one diagonal stripe is created by the string of events which point towards bar 15.

The third section contains the most diagonal features. The four features (D4–D7) are implied by various cluster of islands as indicated on the figure. The diagonal feature D7 is particularly weak and is only suggested by one relatively small cluster of points. The other three features are all supported by at least two distinct islands of points.

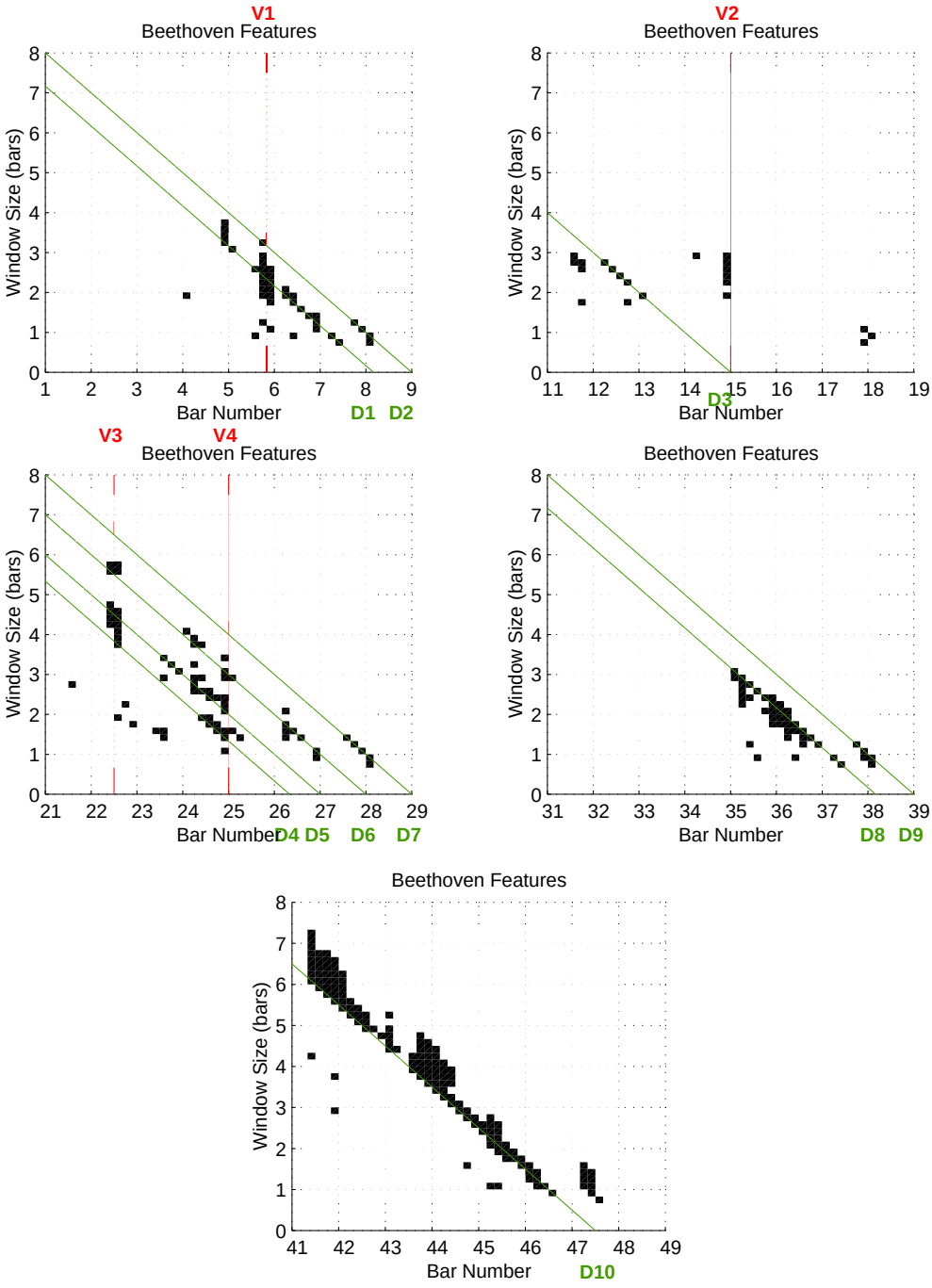


Figure 8.10: The annotated and thresholded curve-fitting results for *Auf dem Hügel sitz ich spähend*.

The two features in the fourth section again vary in strength. The first feature, D8, is implied by the large island centred around $(36, 2)$. This feature is relatively strongly supported as opposed to D9 which is only weakly implied by the island centred at $(38, 1)$.

Finally, the last section contains one very strong diagonal stripe of low errors (D10) which terminates approximately half-way through bar 47. The feature is supported by almost all combinations of window-size and starting position which lie along its principle axis.

Horizontal Patterns

There are no clear horizontal patterns in the thresholded data.

8.5 Results: *Gute Nacht*

The same process of thresholding then looking for patterns was applied to *Gute Nacht*, this is described in the following sections.

8.5.1 Thresholding

Figure 8.11 shows the frequency distributions of the curve-fitting results for the three performed sections of *Gute Nacht*. The distribution pattern as was seen for *Berceuse* is evident. The six peaks, corresponding to the possible combinations of good and bad curve-fits amongst the five pieces, dominate the frequency landscape.

The same threshold of 0.7×10^{-5} was applied across the data to select those good curve-fits which are supported by at least four of the five performances. The following sections describe the features that are present in the thresholded results.

8.5.2 Features in Section One

Figure 8.12 shows the thresholded curve-fitting results for the first section of *Gute Nacht*. There are a large number of well defined features present in the results.

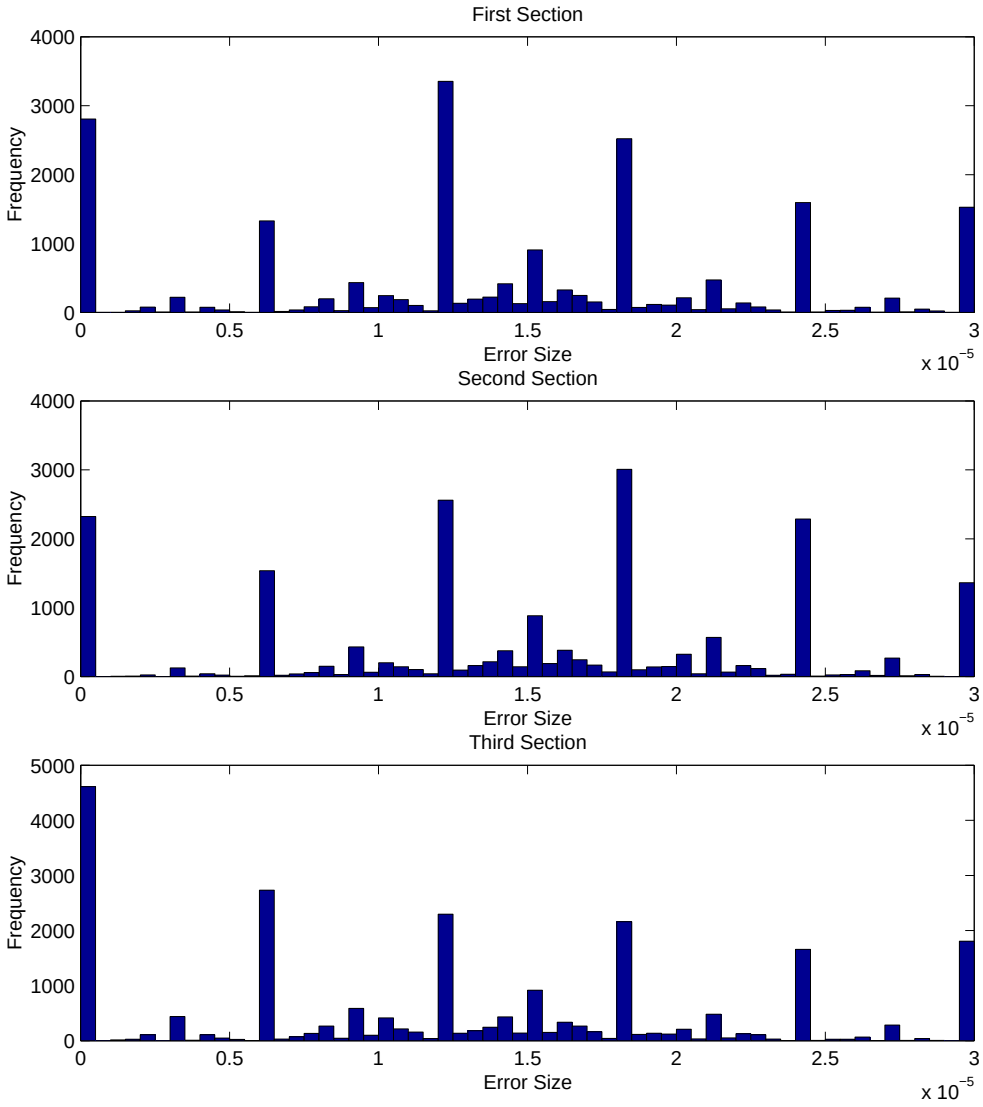


Figure 8.11: Frequency distributions for the curve-fitting results of *Gute Nacht*.

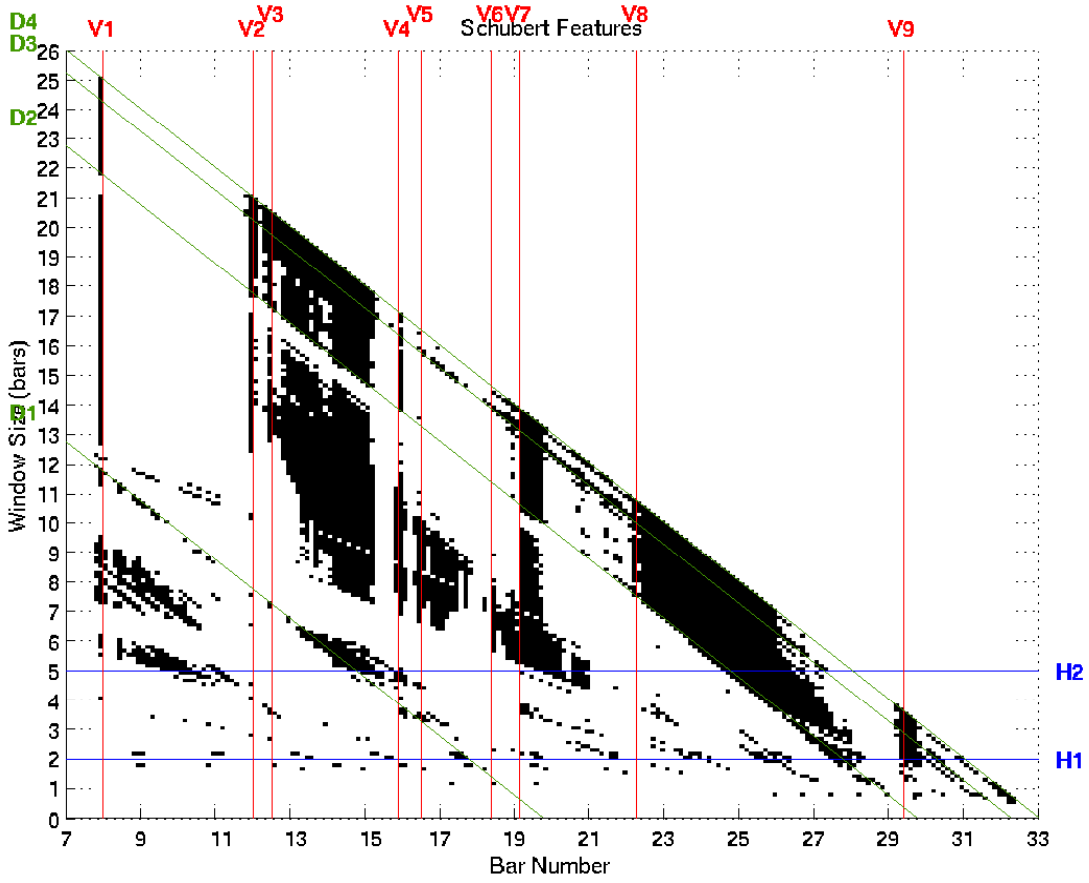


Figure 8.12: Thresholded performance analysis results with annotations for the first thirty-three bars of *Gute Nacht*.

Vertical Patterns

There are nine vertical patterns of features in this first section. The majority of them are caused by distinct vertical lines of low error which clearly stand out from the surrounding details. The features marked V1, V2, V3, V4, V5 and V6 are all examples of these.

The other cause of vertical features are the edges of large islands. V7 is created by the coinciding edges of the two islands at $(20,8)$ and $(20,12)$. The edge of the island at $(24,7)$ creates V8 and, finally, the smaller island at $(30,3)$ creates the last vertical feature (V9).

Diagonal Patterns

Four diagonal features span this first section of the piece. Three of them are clustered near the end of the section, the first terminates near the middle. The first feature, labelled D1, is primarily implied by the edge of the island at $(15,5)$ and supported by the numerous smaller islands that lie along its path.

The second feature (D2) is suggested by the edges of the islands at $(14,18)$, $(20,12)$ and $(24,8)$. The last two diagonal features of this section (D3 and D4) are supported by the three islands above, the interconnections between them and some coinciding strips of islands in the last four bars.

Horizontal Patterns

There are two horizontal features, both of which are interesting in different ways. The first feature (H1) occurs around $y = 2$. The feature is defined by two parallel sets of features with slightly different frequencies which go in and out of phase with one another. These two features straddle the $y = 2$ line and, when they coincide, produce islands that intersect it. It would be highly unusual for two features, which are so similar to one another, to exist in a piece of music and for this reason the two features, are treated as one feature which occurs at $y = 2$.

The other feature is interesting due to its relatively strange frequency. The feature (H2) at $y = 5$ is represented by four large islands of low errors. The size is unusual as

it does not correspond to an obvious structural unit in the piece. The feature may be an artifact created by the two-bar pause between the first four and the last two phrases in this section.

8.5.3 Features in Section Two

The thresholded results from the second section of the piece (see Figure 8.13) show a similarly large set of features as the first section. There are numerous chains of small islands that form distinct vertical and diagonal features. As before, the specific features which have been annotated in the figure are discussed below.

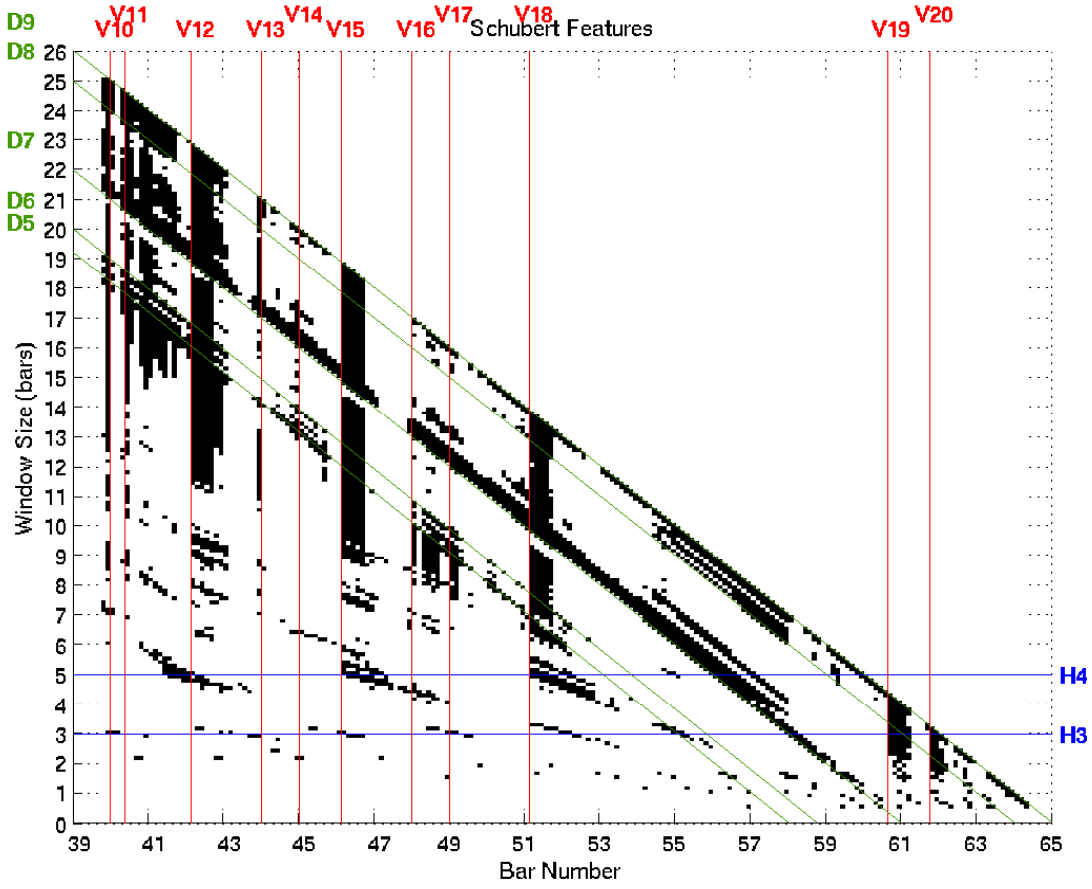


Figure 8.13: Thresholded performance analysis results with annotations for bars 39 to 65 of *Gute Nacht*.

Vertical Patterns

This second section of the piece contains eleven distinct vertical features. Four of these features (V12, V15, V18 and V19) are particularly prominent and are formed by the initial edges of quite large islands terminating at bars 42, 46, 51 and 61 respectively. The other features, although less obvious are still strongly supported, for example features V10, V11 and V13 are created by long stretches of vertical islands that coincide. The remaining four features, V14, V16, V17 and V20, are created by chains of smaller islands.

Diagonal Patterns

The five diagonal patterns have a similar range of strengths. Two of them, D7 and D9, created by large, mostly continuous, chains of islands stretching across most combinations of starting position and window size. The remaining three, D5, D6 and D8 being created by smaller chains of islands which, nevertheless, do extend across most of the figure.

Horizontal Patterns

The horizontal features are again interesting. The same horizontal feature at $y = 5$ remains (labelled as H4) however the feature at $y = 2$ is replaced by a different one at $y = 3$ (labelled as H3). This horizontal feature at $y = 3$ is quite strong and is represented continuously across the figure.

There are remnants of the horizontal feature at $y = 2$ at the start of the section, however this soon fades out within the first 7 bars.

8.5.4 Features in Section Three

Figure 8.14 shows the thresholded results for the final section of *Gute Nacht*.

Vertical Patterns

Similarly to the previous two sections, there are a large number of vertical features present in this section. In this section, a number of the features are less distinct than

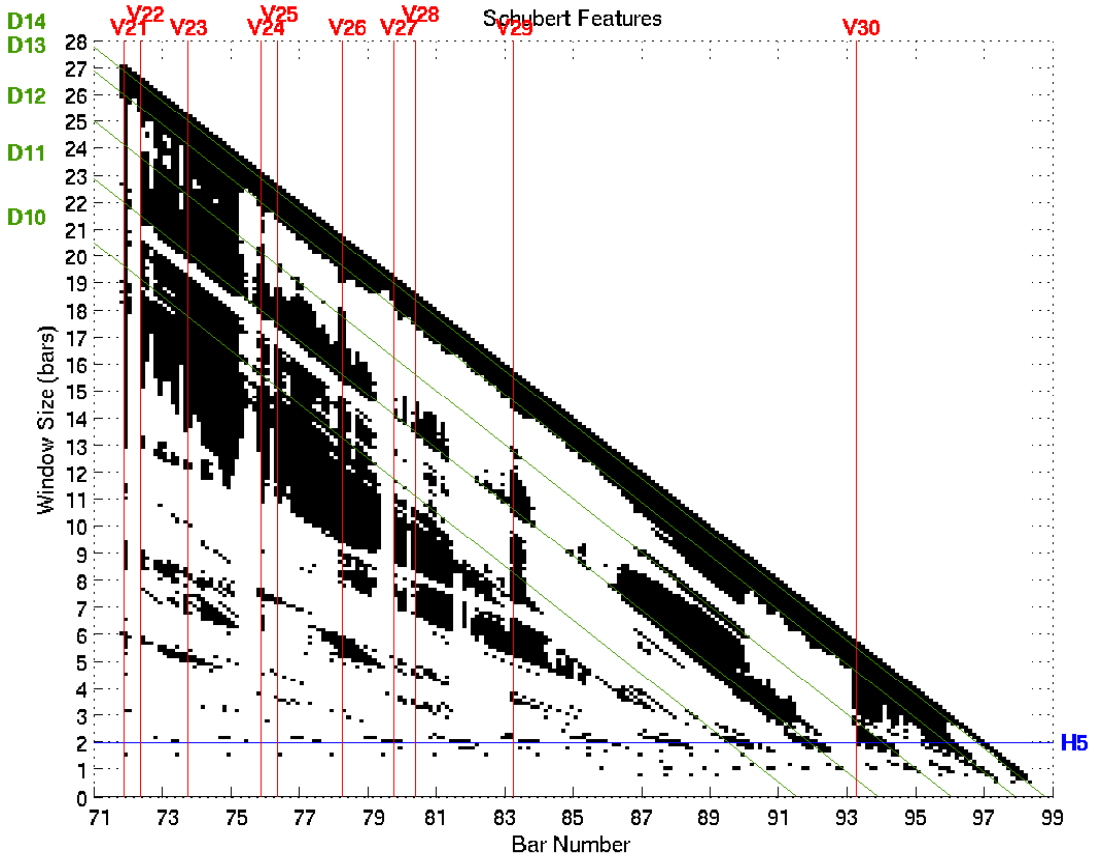


Figure 8.14: Thresholded performance analysis results with annotations for bars 71 to 99 of *Gute Nacht*.

other features as they pass through larger areas of low error. Most of the features are clustered towards the first half of this section with only one feature, V30, lying towards the end of the section.

Diagonal Patterns

There are five clear diagonal patterns in this section of the piece. Four of these patterns, D10–D13, are spaced approximately two-bars apart from each other. D11 and D12 are particularly strong and are supported by most combinations of starting position and window size. The other two features are less strongly supported but are implied by several rows of islands which lie on their path. The final diagonal feature (D14) spans across the entire figure and occurs at the very end of the piece.

Horizontal Patterns

There is one prominent horizontal feature which is reminiscent of the one discovered in the first section. The feature, H5, consists of a line of alternating errors around $y = 2$. As with the first section, there are also two other lines of alternating errors either side of the line at $y = 2$ however in this case, the line is reinforced by points of low error which do occur directly on the line.

8.6 Future Work

As mentioned in the introduction, the results of this chapter have been obtained from a manual analysis of the data. In the future this analysis could be automated. This section provides some suggestions which will aid the development of an automated process.

There are two key steps in analysing the curve-fitting results, removing the ‘noise’ from the results and then identifying the three types of features in the remaining data.

In this chapter, the curve-fitting data was subjected to a threshold to remove occurrences of low-magnitude errors that occurred in less than four of the five performances. This is a reasonable approach to selecting which features are shared across most of the performances, however it is an uninformed approach. A more informed

approach could make use of knowledge of the piece along with knowledge of the kinds of artifacts which may occur when using a curve-fitting process.

Another area of investigation is the automatic detection of the three features. The three features can be classified into two different groups:

Continuous Features correspond to those that are formed by a set of points of low errors lying along a particular straight line. In the case of these analyses, only two angles of lines are of interest, vertical lines and lines at $-45°$ to the horizontal.

Regular Features are those that occur with a specific regularity. For the purposes of this research, this set of features can be reduced to those that produce equi-spaced points of low error where the spaces between the low-errors matches the height on the y-axis at which the error appears.

The precise nature of these features will aid an automated approach to detecting them.

8.7 Summary

This chapter outlined an approach for identifying features which were present in the performance analysis results. The three different kinds of features were introduced along with an explanation of what they represented in the musical performance.

The process to identify these features began with the application of a threshold to the raw data to reduce the relatively complex set of data to a simpler form which would be simpler to work with. The thresholding value was strictly set to include only those points of low curve-fitting error which occurred in four out of five performance analyses.

Once the data had been reduced, the individual sections of the pieces were plotted and visually examined for the three features described earlier. For the purposes of this work, the process of inspecting the data for features was conducted manually but a necessary further step would be to automate this process.

The identified features were labelled and can now be applied to the results of the grouping analysis in order to confirm or reject some of the grouping boundaries. This

is the subject of the next chapter.

Chapter 9

Synthesis

9.1 Introduction

The research presented in previous chapters has provided two sources of information about the musical structure of the pieces. Chapters 4 and 5 presented an implementation of Lerdahl and Jackendoff's Grouping Structure and a means of representing the resulting grouping structure which supports a process of gradual refinement. The refinement process is controlled by a set of switches which are placed on each possible grouping boundary (see Section 4.5.4). If the switch is on, the grouping boundary is considered active and the relative strength of that boundary is used to control the higher level shape of the grouping structure. If the switch is off, the grouping boundary is considered inactive and the neighbouring notes, which were potentially separated by that boundary, will be collected into the same group.

The research in Chapters 6 and 7 produced a method of analysing a musical performance in order to detect occurrences of phrase-final lengthening. The results from this analysis process are a collection of features in the performance which are assumed to correspond to features of the musical structure. The location of these features are key in determining whether a boundary should be active or inactive.

The work presented in this chapter joins these two separate analyses together (see Figure 9.1) and compares the results of this joint analysis with the results of applying the L&J analysis on its own. The features discovered in the musical performances are

used to determine which of the boundary points should be considered active and which should be inactive.

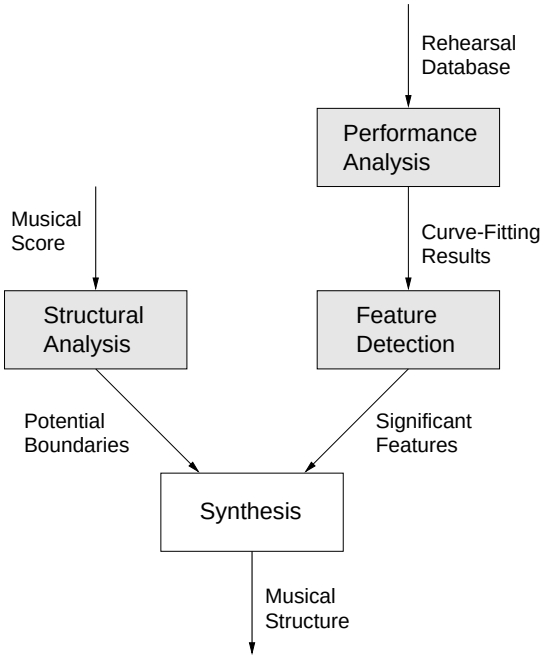


Figure 9.1: Pictorial representation of the rôle of this component within the structural disambiguation flow.

The chapter begins with an overview of the process which joins the results of the structural and performance analyses.

9.2 Synthesis Process

The rôle of the synthesis process is to combine the results of the structural and performance analyses. The structural analysis produces a set of possible grouping boundary points with associated weights and switches. The performance analysis produces a set of features which correspond to interesting patterns in the musical performances.

The process described below takes the output from each of these analyses and finds structural features that are common to both analyses. Specifically, the research in this chapter matches the features discovered in Chapter 8 with the switches controlling the

potential grouping boundaries.

As with the previous chapter, for the purposes of this research, the process was conducted manually by trying to find occurrences of features which intersected with the previously identified grouping boundaries.

Two different approaches were needed, one for the horizontal features and another for the vertical and diagonal features.

9.2.1 Horizontal Features

As described in the previous chapter, a horizontal feature consists of a number of islands which lie at the same vertical height and which are separated by a fixed amount. The horizontal features, once identified, have to then be converted into corresponding points on the x-axis in order to influence the choice of boundary points. These intersections with the x-axis should be chosen to reflect the nature of the horizontal features i.e. they should be equispaced at a distance corresponding to the height of the horizontal feature.

In general, the horizontal features were mapped to the x-axis at the point corresponding to the leading edge of the islands of which the horizontal feature consisted. In the more complicated case of a chain of islands of multiple sizes, the x-axis intersections were chosen so that the intersections aligned to the maximum number of islands on that feature. For example, feature H6 in Figure 8.7 was chosen to intersect the x-axis at the cusp of every second bar even though the island at bar 43 starts halfway through bar 42.

9.2.2 Vertical and Diagonal Features

Algorithm 9.1 shows a boundary selection algorithm which would work under perfect conditions for vertical and diagonal features. The variables $location_f$, $location_{a,b}$ and $location_{c,d}$ refer to the location in time of the feature (f) under consideration (i.e. where it would intersect the x-axis), the nearest boundary occurring immediately before f and the boundary occurring immediately after f. The *str* and *sw* variables refer to the boundary strength and the state of the switch at the location indicated by their

subscripts.

Algorithm 9.1 Boundary selection algorithm.

BOUNDARYSELECTOR($location_f$, $location_{a,b}$, $location_{c,d}$)

1: **if** $\left(\left(location_f - location_{a,b}\right) = \left(location_{c,d} - location_f\right)\right)$ **then**
2: **if** $\left(str_{a,b} > str_{c,d}\right)$ **then**
3: $sw_{a,b} = 1$
4: **else**
5: $sw_{c,d} = 1$
6: **end if**
7: **else if** $\left(\left(location_f - location_{a,b}\right) < \left(location_{c,d} - location_f\right)\right)$ **then**
8: $sw_{a,b} = 1$
9: **else**
10: $sw_{c,d} = 1$
11: **end if**

The algorithm inspects the distances between the discovered feature and the nearest surrounding boundaries. If the feature is equidistant between the surrounding boundaries then the algorithm would choose the feature which has the greatest strength.

However, given the varying note density throughout a piece and the fact that boundary points are specified as occurring between two events, it is unsatisfactory to merely use the linear distance between the boundaries and the feature as a precise deciding factor.

When performing the mapping process, an exact match of feature to grouping boundary was not required because, by their nature, the grouping boundaries occur during a period of musical time and so a precise location for them cannot be given. However, a feature was only considered to coincide with a grouping boundary if it was close to that boundary. The precise definition of ‘close’ would have to be derived from a larger set of data; for this research, a feature was considered close to a boundary based on the particular boundary’s context. For example, for musically dense sections of the pieces, a feature/boundary match would only occur if the feature indicated the correct section of the relevant bar. For less dense sections, such as rests, the feature/boundary match was given a greater degree of freedom.

Another heuristic used to decide which boundary to select was based upon the bar in which the feature occurred. So, for example if a feature lay equidistant between two boundaries but was in the same bar as one of those boundaries then the tendency was to apply the feature to the boundary in the same bar.

The following sections present the results of applying the synthesis process to the three pieces used in the empirical study (Chapter 6).

9.3 Synthesis: *Berceuse*

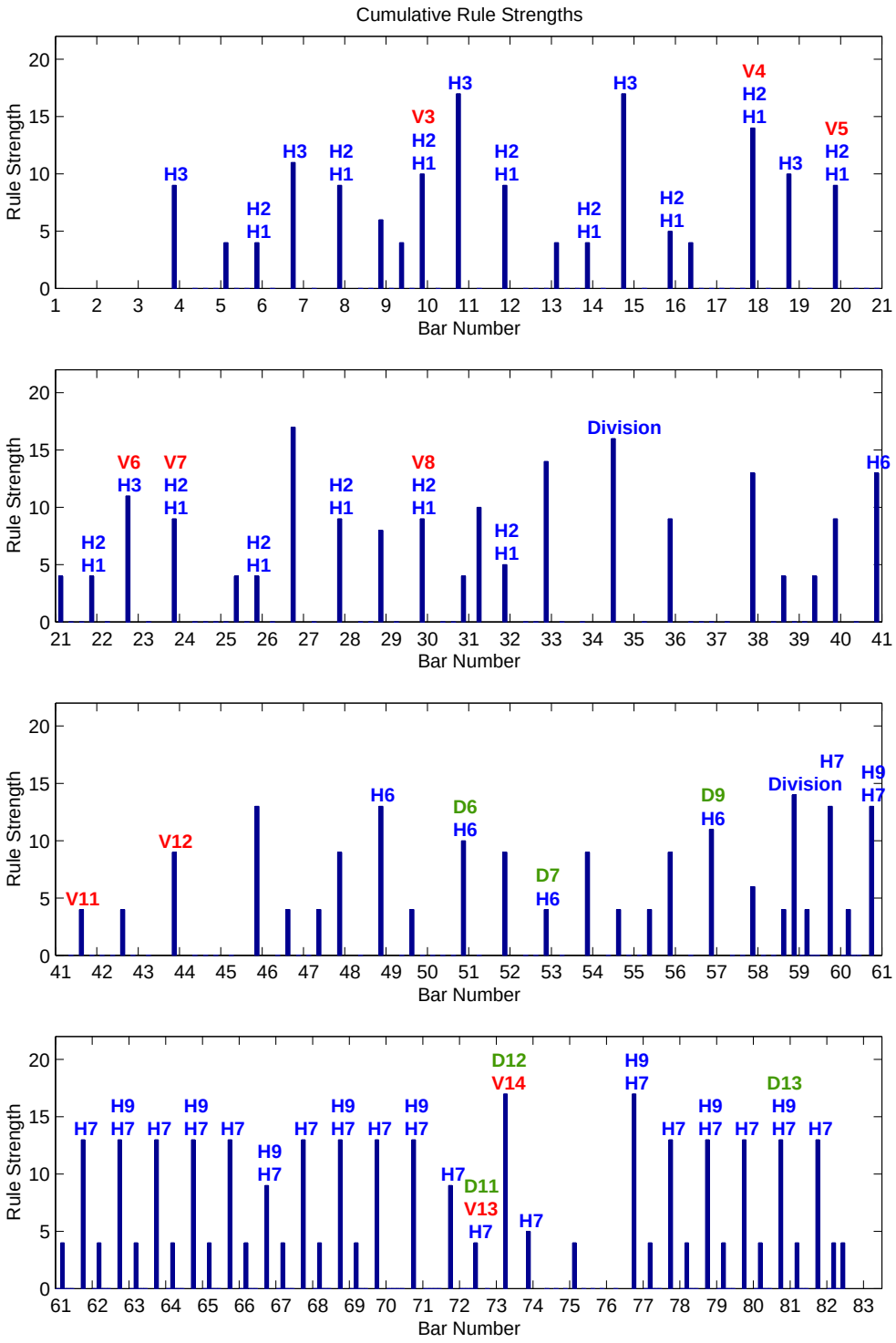
Figure 9.2 shows the set of possible grouping boundaries as derived by the implementation of L&J's Grouping Rules. The height of the bars of the graph indicate the strength of the potential boundary at that point (based on the chosen feature-category weighting factors). The annotations above the bars indicate where features identified in the performance intersect with the possible boundary points.[1] If a bar has no annotation near the top of it then there was not a corresponding feature identified in the performance and so the switch representing that boundary would be made inactive.

Forty-nine (50%) of the remaining ninety-eight[2] possible boundary positions have coinciding features in the performance analysis. The twenty-two features from the performance analysis which match boundary positions, are in turn approximately 59% of the total number of features found in the performance analysis. In other words, approximately half the boundary positions are selected using the performance features and almost two-thirds of the performance features correspond to possible boundary points.

Of more importance than the number of intersections between the identified features and the structural analysis is whether these intersections occur at the correct places. Specifically do those intersections select those boundary points which correspond to the musical structure of the piece. The following section evaluates the resulting grouping structure.

[1] The annotation of Division on some of the bars indicates those boundary points around which the piece was divided into sections and are assumed to be both active and stronger than the boundary points between them.

[2] After the division boundary points are excluded.


Cumulative Rule Strengths
Rule Strength
Bar Number
H3
H2 H1
H3
H2 H1
V3 H2 H1
H3
H2 H1
H3
H2 H1
H2 H1
H3
V4 H2 H1
H3
V5 H2 H1
V6 H3
V7 H2 H1
H2 H1
H2 H1
H2 H1
V8 H2 H1
H2 H1
Division
H6
V11
V12
H6
D6 H6
D7 H6
D9 H6
Division
H7
H9 H7
H7 H9 H7 H7 H9 H7 H7 H9 H7

9.3.1 Evaluation

There are two aspects of the evalution:

- How does the resulting grouping structure relate to the music?
- How and where does the resulting grouping structure benefit from the input from the performance data?

The first of these questions is dealt with by discussing what features of the music the analysis did and did not capture. The second question is answered by comparing the results of the informed grouping structure with the grouping structure which would result from only applying the grouping rules.[3]

In order to evaluate the informed grouping structure, each section of the piece is taken in turn and a comparison between the grouping structure resulting from the synthesis and the structure implied by the score is discussed.

First Section The results for the first section of *Berceuse* (as shown in Figure 9.3) show that the selected boundaries capture the majority of the structural features of the excerpt. For example:

- The analysis clearly identifies the four-bar phrase structure of this first section;
- Within each four-bar phrase, the analysis identifies the smaller scale structure;
- Events e043 and e044 from the end of the fourth four-bar phrase are grouped with the subsequent four-bar phrase against the regular structure implied by the previous phrases but in line with the performance and the score;
- Approaching the end of the section, the analysis detects the change in musical structure.

However, there are also a number of points where the analysis does not agree with the structure implied by the piece. For example:

[3]This second analysis, obtained by just applying the grouping rules, will be referred to in the remaining text as the *vanilla analysis* of the piece. Whereas the analysis including performance data will be referred to as the *informed analysis.*

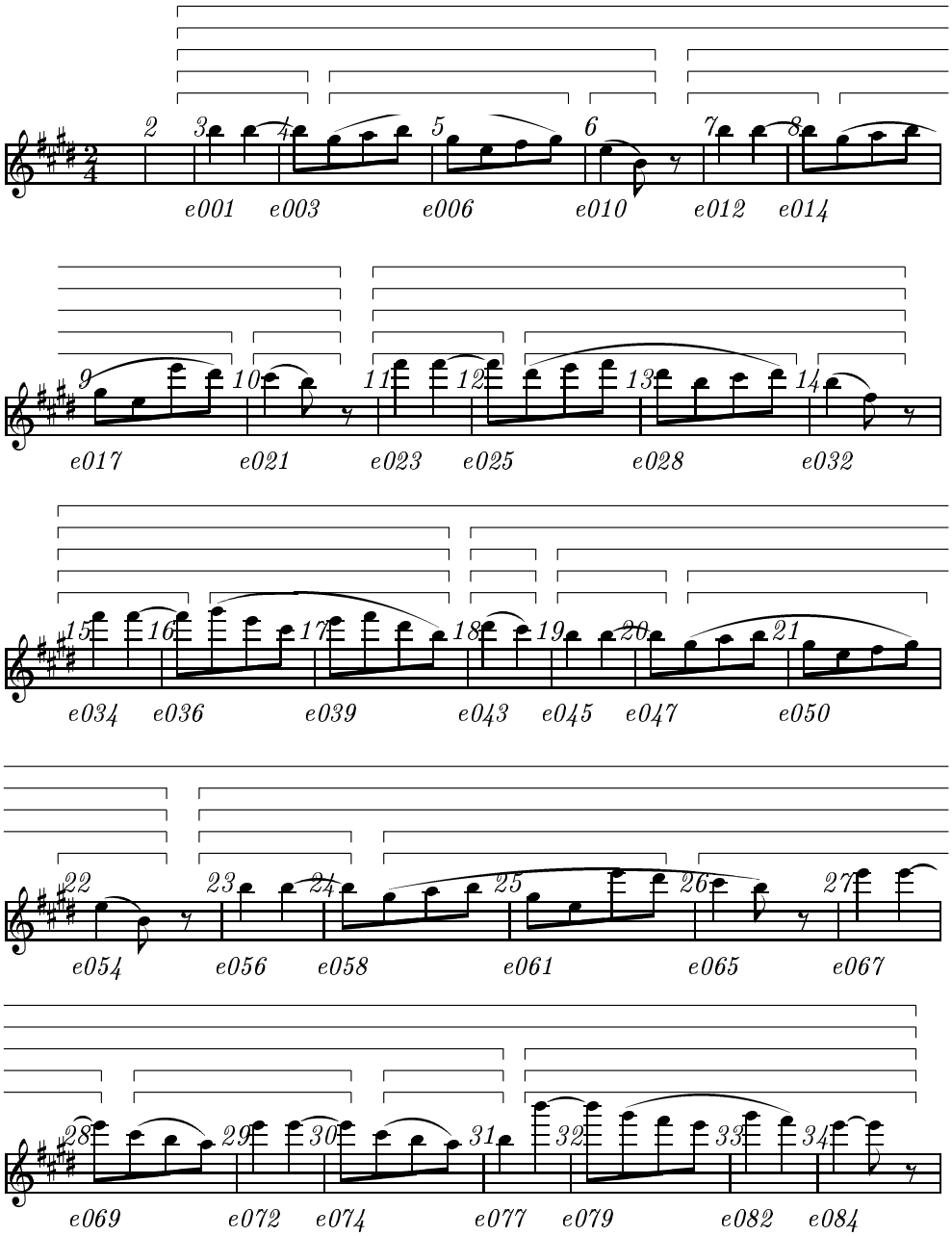


Figure 9.3: Grouping structure for the first third of *Berceuse* arising from the combination of the performance and structural analyses.

- Between events e018 and e019 (and similarly between e062 and e063) there should probably be a boundary introduced by the large jump in register;
- The boundary between e033 and e034 is too strong and divides the first section inappropriately;
- Between events e071 and e072 there should probably be a boundary.

Of these problem cases, the first one is identified by the grouping analysis but not supported by features in the performance analysis. The second problem is due to the feature-category weightings assigned to the rules that apply at that grouping boundary. The last of these cases is supported by the grouping rules, however it is not clear from the performance data whether the boundary should be supported or not.

Figure 9.4 shows the results of taking the vanilla analysis of the piece and selecting all the grouping boundaries which have a strength greater than four. A threshold was applied to the vanilla analysis because selecting all the grouping boundaries produced a hierarchy that was over–crowded and had many singleton events. A threshold of four removed these singleton events and produced a hierarchy that, for the most part, agrees with the music.

The results from the thresholded vanilla analysis are very similar to those from the informed analysis with only eight differences between the two analyses.

- The informed analysis correctly detects the lower level phrase structure near the end of the four-bar phrases (after events e009, e031, e053, e064) which are missed by the vanilla analysis;
- Between events e016 and e017, the vanilla analysis produces a weak boundary which is not supported by the music nor the performance;
- The informed analysis misses an important boundary point after event e066 which is detected by the vanilla analysis;
- The vanilla analysis incorrectly creates a singleton group for event e078;
- Between events e081 and e082, the vanilla analysis produces a strong grouping boundary which is not supported by the music nor the performance.

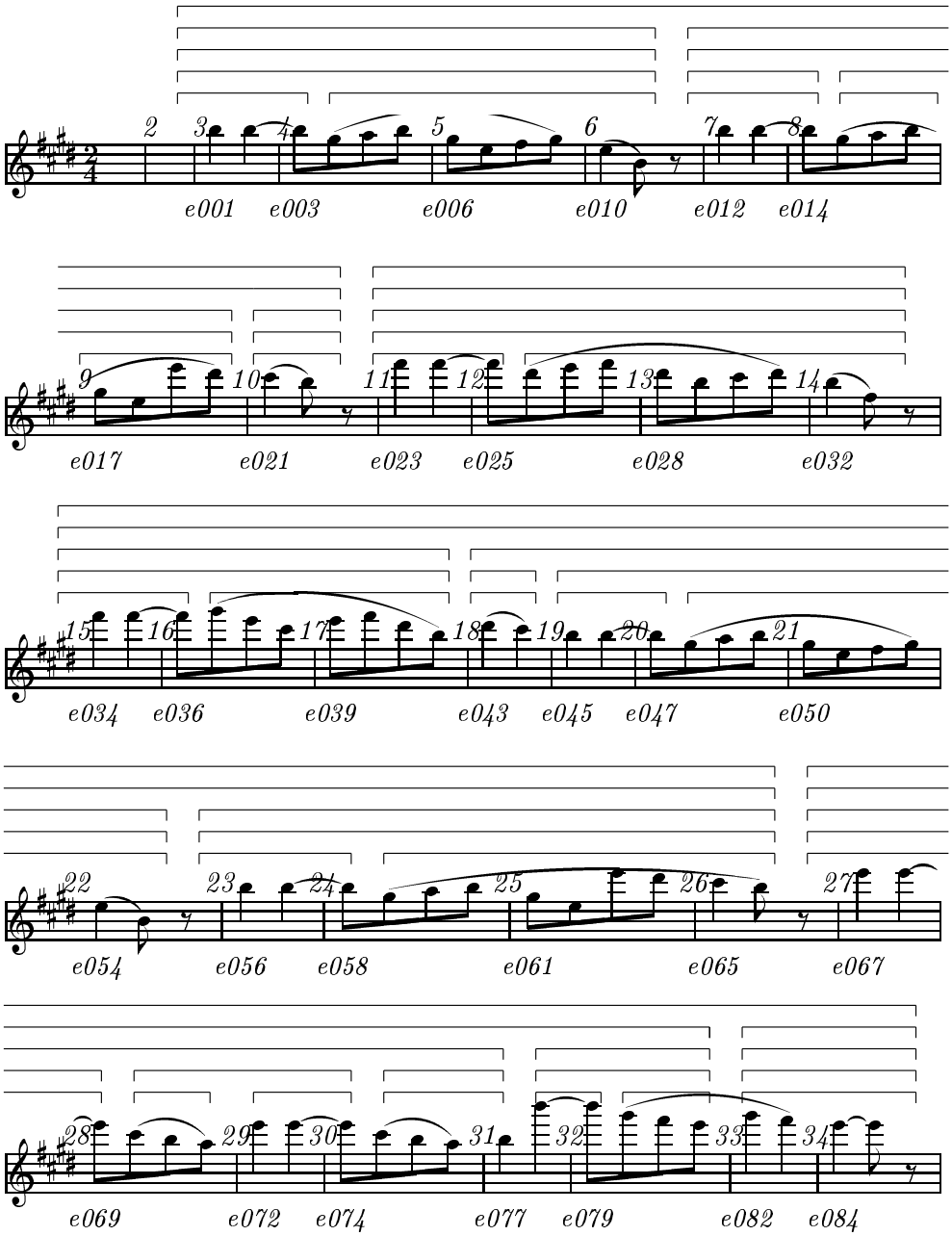


Figure 9.4: Grouping structure for the first third of *Berceuse* arising from the thresholding the structural analysis results.

Finally, some of the issues that were flawed in the informed analysis remain in the vanilla analysis: e.g. the lack of a boundary after events e018 and e062 and the boundary that is too dominant after e033. However the missing boundary after e071 is present in the vanilla analysis.

Second Section The second section is the least rigorously structured of the three sections, a fact that is apparent both in the results of the grouping analysis and the results of the performance analysis. Figure 9.5 shows the results of combining the two analyses.

The results of the synthesis for this section of the piece are weak. The L&J grouping analysis correctly detects the two-bar phrase structure for this section. This two-bar structure is also identified by the performance analysis module however the module's results are one-bar out of phase with the grouping analysis. So although both techniques discovered a two-bar structure, because the occurrences do not coincide, the final structure is inaccurate.

Figure 9.6 shows the results of thresholding the vanilla analysis. For this section of the piece, the resulting grouping does correcting reflect music with only one questionable grouping boundary after event e128.

Final section The results from the final section of the piece are again encouraging (see Figure 9.7). The analysis successfully detects the one-bar repeating phrase structure which constitutes the majority of this final section. The change in musical structure when the repetition of the four-bar phrase occurs is also detected.

However, there are two issues with the final structure:

- The intermediate levels of the grouping structure for bars 59–73 do not reflect the structure of the music;

- Even though the four-bar phrase is treated differently from the surrounding one-bar phrases, the internal structure of the phrase does not match those of the four-bar phrases detected earlier. This can partially be explained by the fact that although the four-bar phrase is clearly similar to the original phrases, the musi-

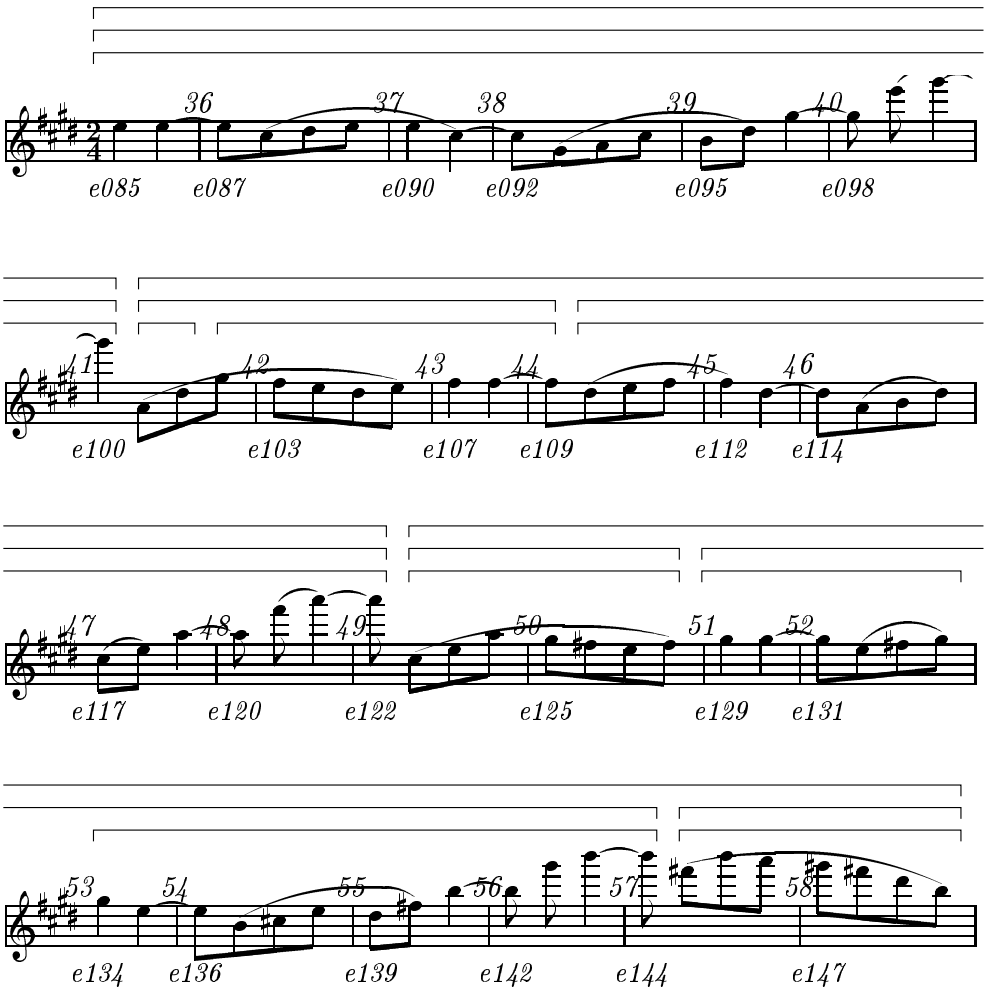


Figure 9.5: Grouping structure for the middle third of *Berceuse* arising from the combination of the performance and structural analyses.

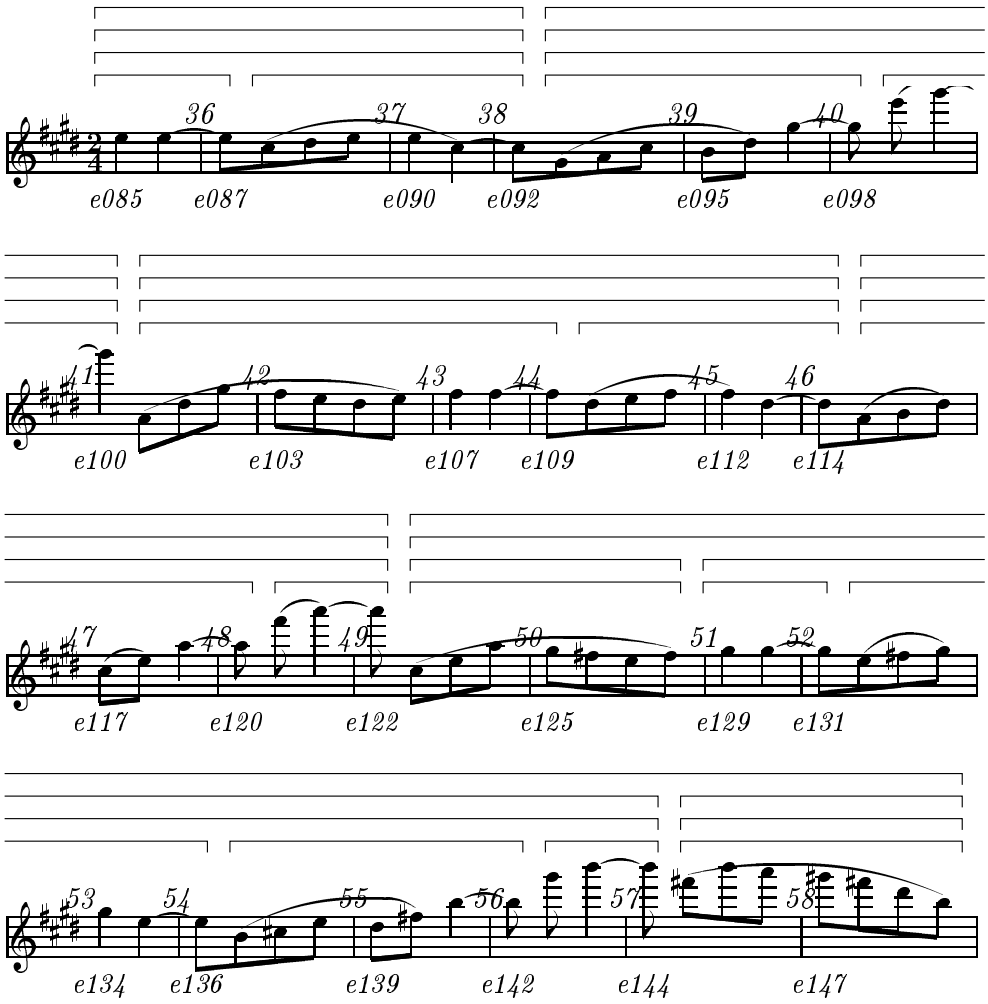


Figure 9.6: Grouping structure for the middle third of *Berceuse* arising from the thresholding the structural analysis results.

cal score (specifically the rest between the initial two notes and the changes in register) does not exactly match the previous occurrences.

Figure 9.8 shows the results of the thresholded vanilla analysis. The two structures are very similar with only one difference occurring immediately after e219. This boundary, created in the informed analysis, does not agree with the music.

9.3.2 Summary

The synthesis of the structural and performance analyses has successfully produced a structural representation for most of the piece. The synthesis was particularly successful during the first and last sections of the piece. Although the synthesis was unsuccessful for the second section of the piece, it is clear from the individual analyses that they both detected aspects of the structure but did not combine them effectively. Perhaps this could be resolved with a more intelligent approach to joining the results of the two analyses.

When comparing the results from the informed analysis against the vanilla analysis, a similar pattern of results was obtained. The informed analysis did particularly well in the first section by identifying part of the recurring phrase structure which the vanilla analysis missed and by correctly not activating a strong boundary identified in the vanilla analysis. In the central section of the piece, the vanilla analysis did extremely well whilst the informed analysis performed badly due to the phase problem. In the final section of the piece, both analyses produced very similar results with one error appearing in the informed analysis.

Section 4.5.2 stated that only a subset of the rules were implemented and that some of the other rules such as GPR6 (Parallelism) would implicitly arise from the performance analysis. Examining the results from the synthesis process, it is clear that similar phrases within the piece did indeed get assigned similar structural analyses.

Overall, the performance analysis has correctly selected many of the grouping boundary points for the first and last sections but failed on the middle section.

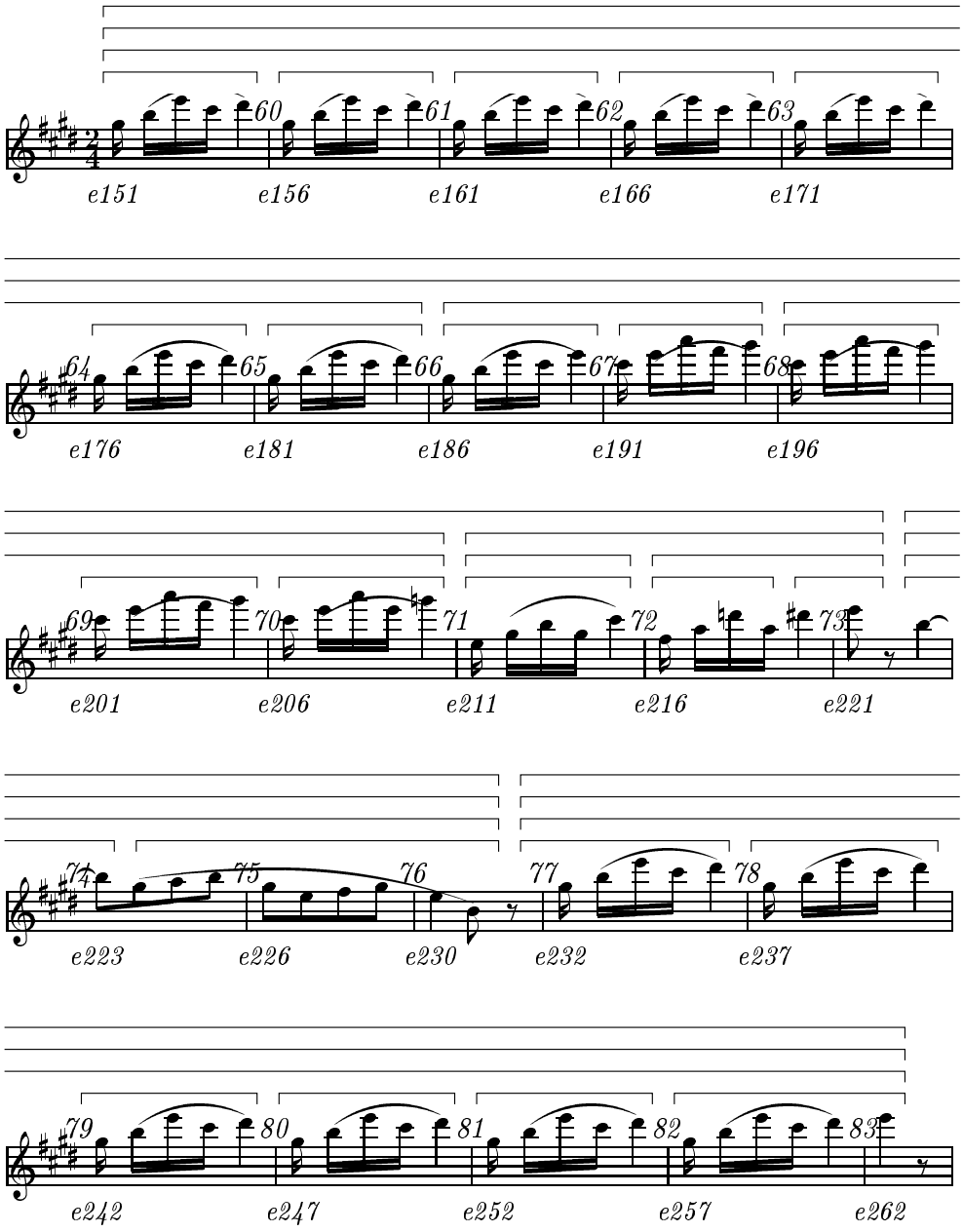


Figure 9.7: Grouping structure for the final third of *Berceuse* arising from the combination of the performance and structural analyses.

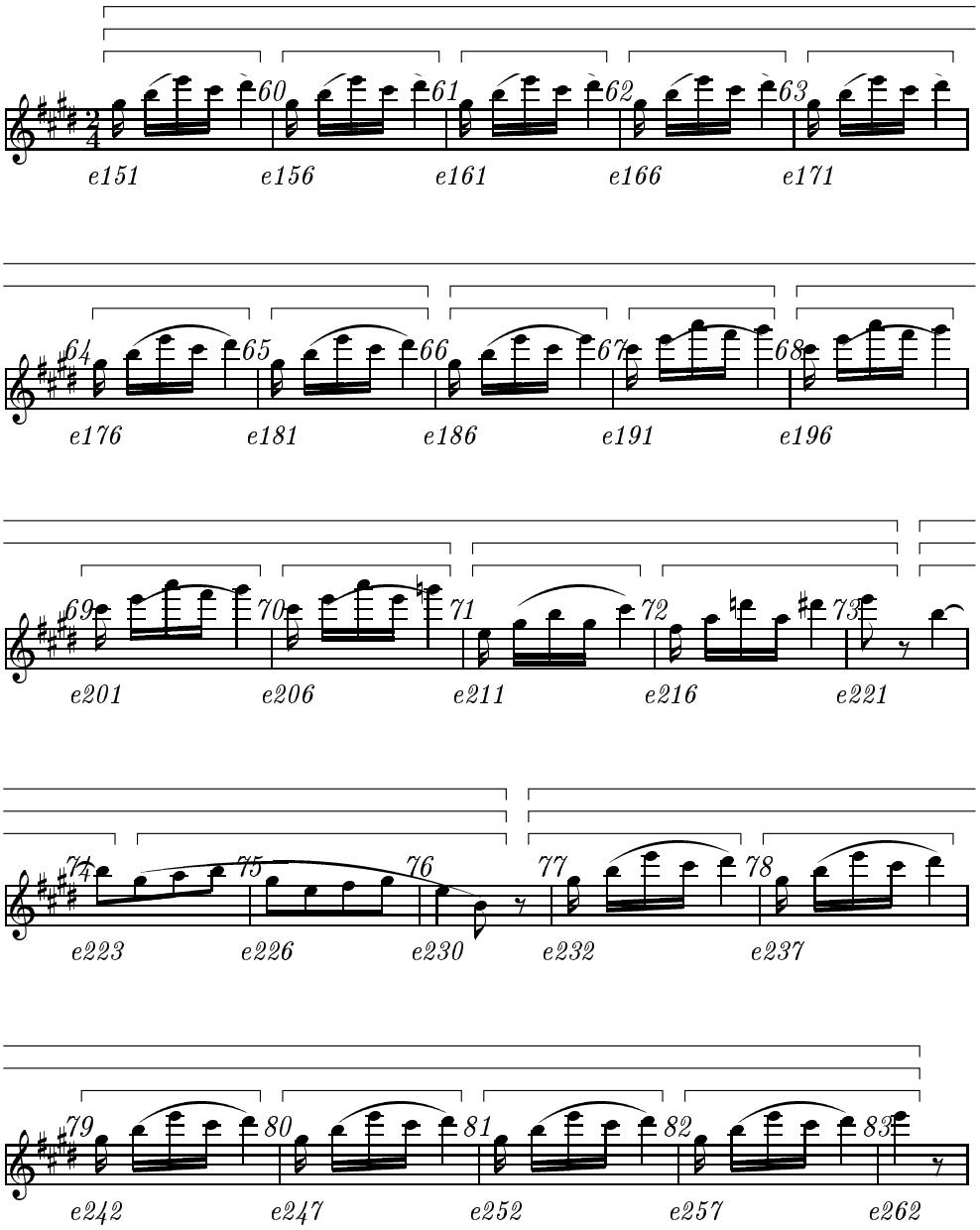


Figure 9.8: Grouping structure for the final third of *Berceuse* arising from the thresholding the structural analysis results.

9.4 Synthesis: *Auf dem Hügel sitz ich spähend*

Figure 9.9 shows the results of joining the performance features with the grouping analysis. As can clearly be seen, very few of the potential boundaries have been selected due to the weak results from the performance analysis.

9.4.1 Evaluation

The effects of converting the active boundaries into a grouping structure can be seen in Figures 9.10 and 9.11.

Figure 9.10 shows the resulting grouping structure for the first three stanzas of *Auf dem Hügel sitz ich spähend*. In the first stanza, only two of the possible thirteen boundary points are selected by the results from the performance analysis. The first of these points (between events e016 and e017) does not reflect the correct grouping boundary which should lie between events e015 and e016. However, the GTTM structural analysis of the piece does not propose this transition as a potential boundary due to the relative instability of the surrounding events.

The second stanza has only one confirmed grouping boundary, as with the first stanza, the boundary which is selected is not the one which is apparent when listening to the music. In this case, the boundary occurs immediately before the correct boundary.

The third stanza presents the most hierarchical information with four selected boundaries. The first of these boundaries (between e065 and e066) does match a grouping boundary which is heard in the music. The second boundary, between e077 and e078, as with the previous two examples, is the wrong boundary for that section of the piece. The third boundary, does constitute a valid possible boundary, however there are a number of surrounding boundaries which are not selected. Finally the last boundary is again potentially correct, however it causes a singleton group and should be trimmed from the final analysis to conform to L&J's grouping structure.

Figure 9.11 shows the grouping structures for the final two stanzas of *Auf dem Hügel sitz ich spähend*. The fourth stanza has two grouping boundaries identified near the end of the section. The first of these, between e122 and e123, does not match a

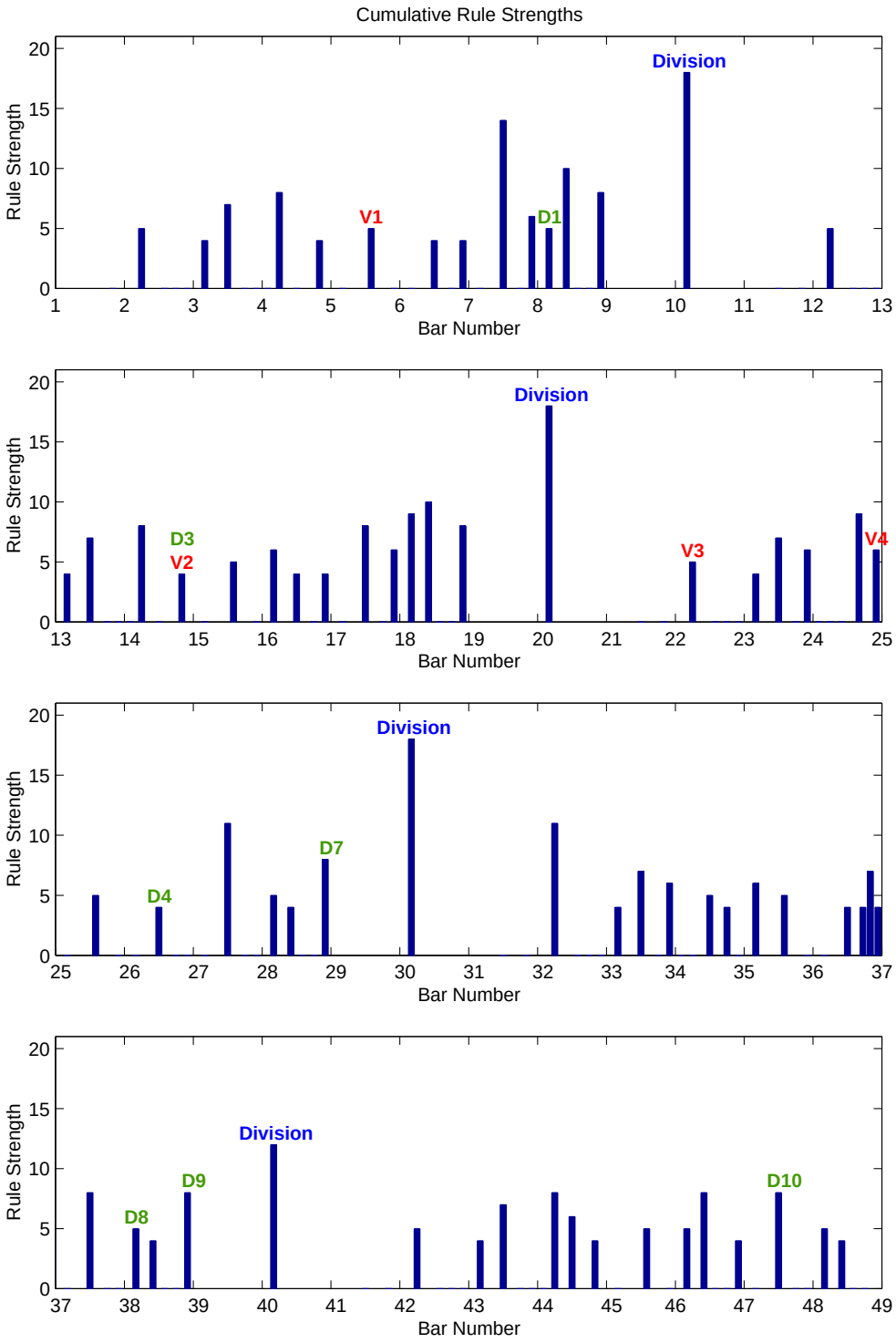


Figure 9.9: Possible grouping boundaries for *Auf dem Hügel sitz ich spähend*

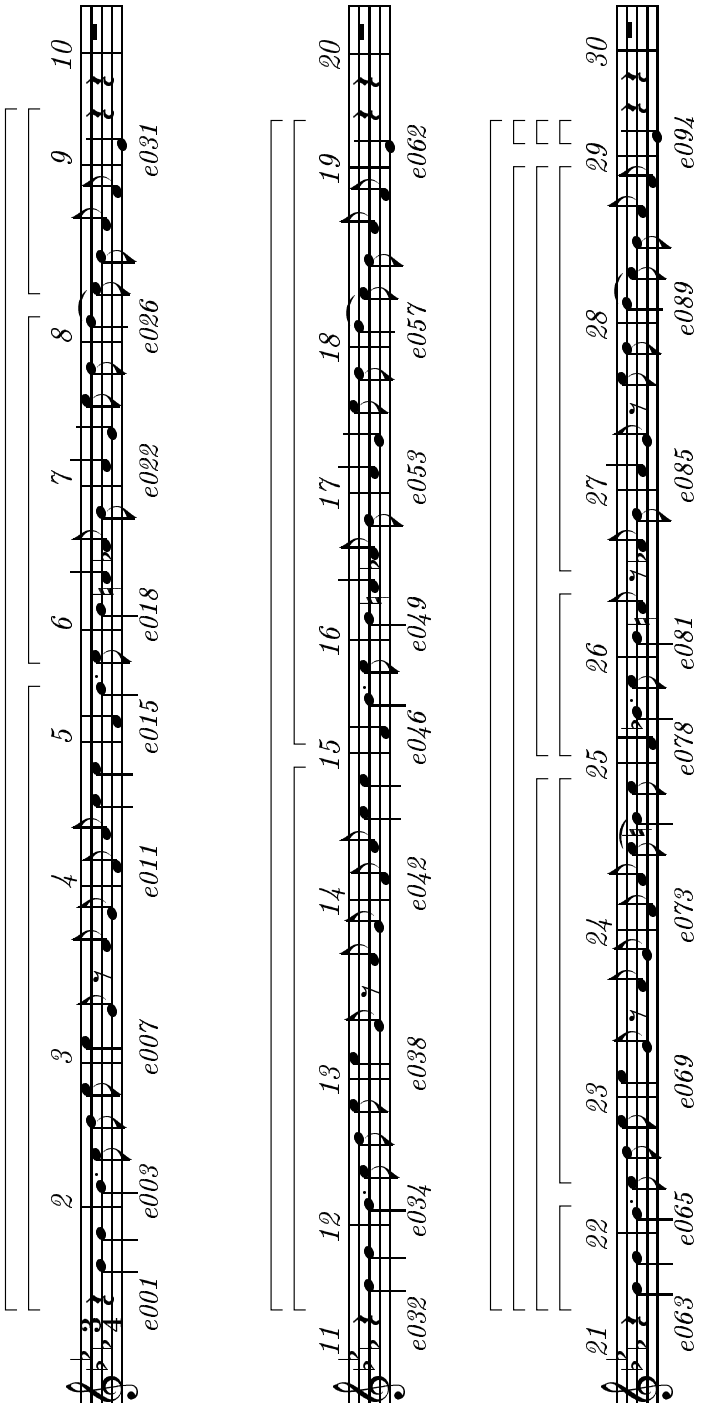


Figure 9.10: Grouping structure for the first three stanzas of *Auf dem Hügel sitz ich spähend* arising from the combination of the performance and structural analyses.

valid boundary. The boundary suggests that two events, which are clearly indicated as belonging together in the score with a slur, should be separated. The final boundary between events e126 and e127 is again not a valid boundary as it incorrectly separates the last event of a phrase from the others belonging to that phrase.

The final stanza has only one grouping boundary (between e151 and e152). This boundary does coincide with the boundary indicated by the score and lyrics which mark the start of the last phrase in the piece.

Figures 9.12 and 9.13 show the results of the thresholded vanilla analysis for the five stanzas of *Auf dem Hügel sitz ich spähend*. The thresholded analysis correctly models the first three bars of each stanza but fails to capture the important boundary between e015 and e016, and the equivalent in each stanza. The analysis does correctly capture the boundary in the seventh bar of each phrase but then produces strong hierarchical features in the last two bars which are not implied by the music.

Overall, there are a large number of boundary points generated which do not correspond to the phrase structure. These can not be removed by increasing the threshold as that will also remove the few valid boundaries that are detected.

9.4.2 Summary

The synthesis process performs poorly for this piece. The grouping structures which are returned by the process do not accurately reflect the structure inherent in the piece. There are two immediate reasons why this may be the case. Firstly, of the three performances, this was the piece that had the greatest variation across its five performances, a fact that was evident from the performance analysis. This may mean that in choosing a threshold that would produce the desired amount of features for the synthesis, the threshold was lowered too much which caused some of the synthesis to be based on either very weak features or noise. Another possible reason may be due to the properties of the piece. The score is marked as *lento ed espressivo* which would probably increase the likelihood of variation across performances. The phrase structure of the piece is also relatively unusual as the phrases begin and end in the centre of the bars which may also affect the performance characteristics.

When the vanilla analyses are examined, they also only capture some of the group-

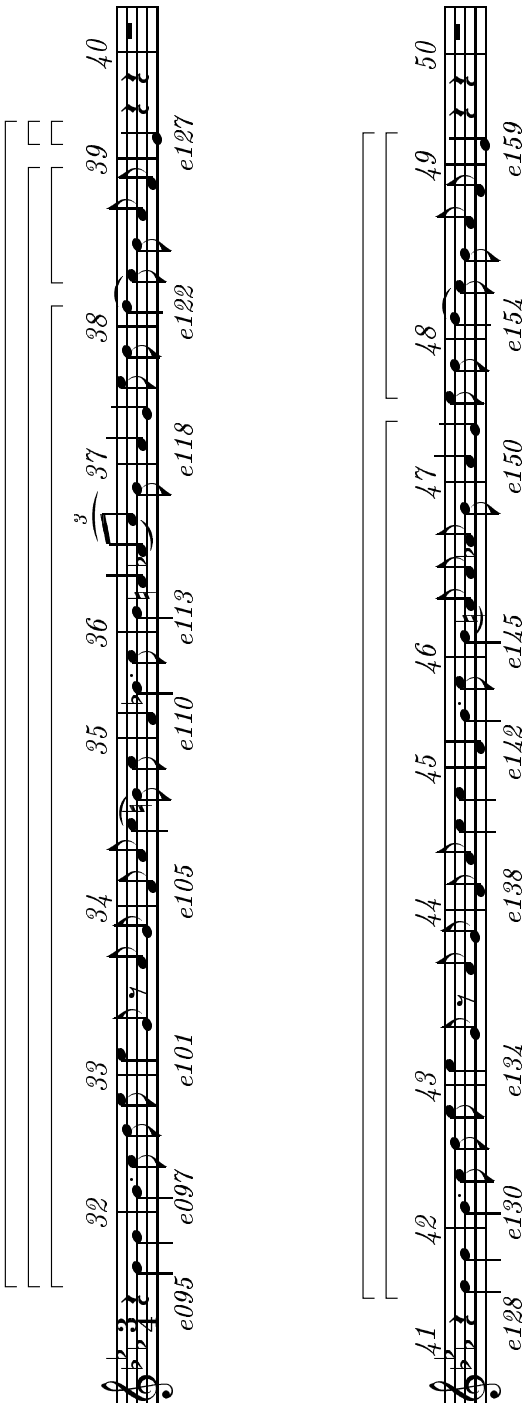


Figure 9.11: Grouping structure for the final two stanzas of *Auf dem Hügel sitz ich spähend* arising from the combination of the performance and structural analyses.

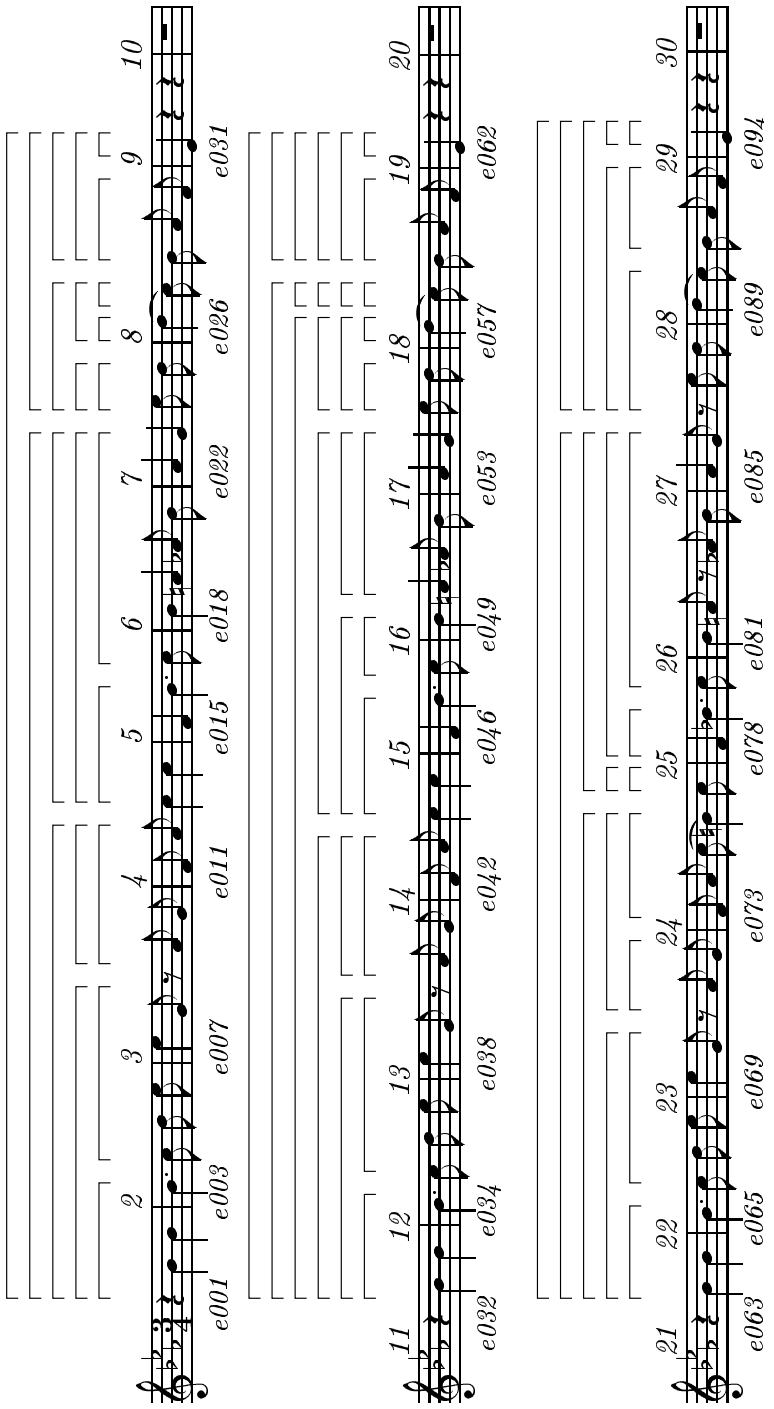


Figure 9.12: Grouping structure for the first three stanzas of *Auf dem Hügel sitz ich spähend* from the thresholded structural analysis.

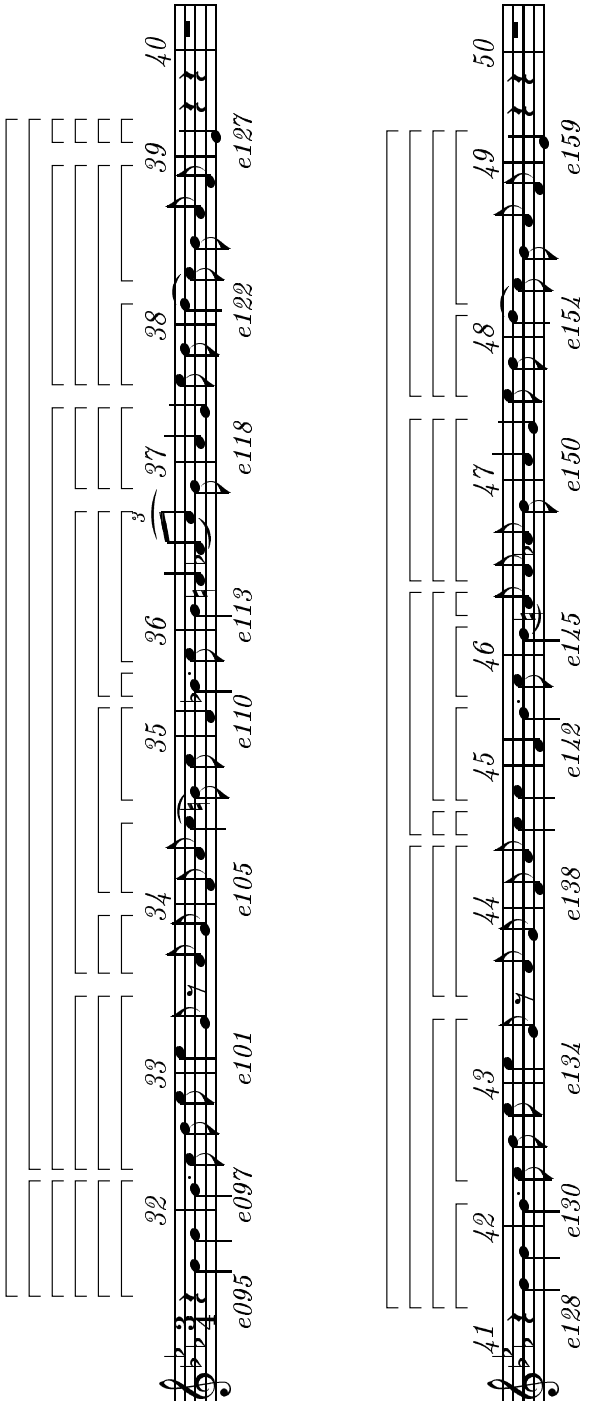


Figure 9.13: Grouping structure for the final two stanzas of *Auf dem Hügel sitz ich spähend* from the thresholded structural analysis.

ing structure of the piece. The grouping rules fail to identify the important boundary in the fifth bar of each phrase due to the relative instability of the surrounding events. There are also a large number of incorrect boundary points generated for this simple phrase structure. However they cannot be eliminated by increasing the threshold because some of the valid boundary points would then be lost.

For this piece, both analyses fail to capture the musical structure accurately however the vanilla analyses does perform slightly better by at least consistently finding several valid grouping boundaries.

9.5 Synthesis: *Gute Nacht*

The grouping boundaries detected by the structural analysis are shown, along with any intersecting performance features, in Figure 9.14. The frequency and distribution of the potential boundaries clearly show that each of the three sections shares a similar underlying structure.

There are one-hundred and ten possible grouping boundaries, of which fifty-four (49%) are confirmed by features in the performance analysis. These fifty-four performance features which have corresponding structural boundaries account for 78% of the total number of performance features detected.

9.5.1 Evaluation

Figures 9.15–9.19 shows the resulting grouping structure that is produced by the synthesis of the performance and structural analyses of *Gute Nacht*.

First Section Figure 9.15 shows the grouping structure for the first third of the piece. The piece is divided into two sections by the large boundary between events e016 and e017. This strong division is not an accurate reflection of the piece in which each of the phrases has similar importance.

As described in Chapter 6, each section consists of three motifs played twice as *aabbcc*. The first pair of motifs in this section *aa* occurs as e001–e016 and e017–e032. The structure clearly indicates the separation between these two phrases (between e016

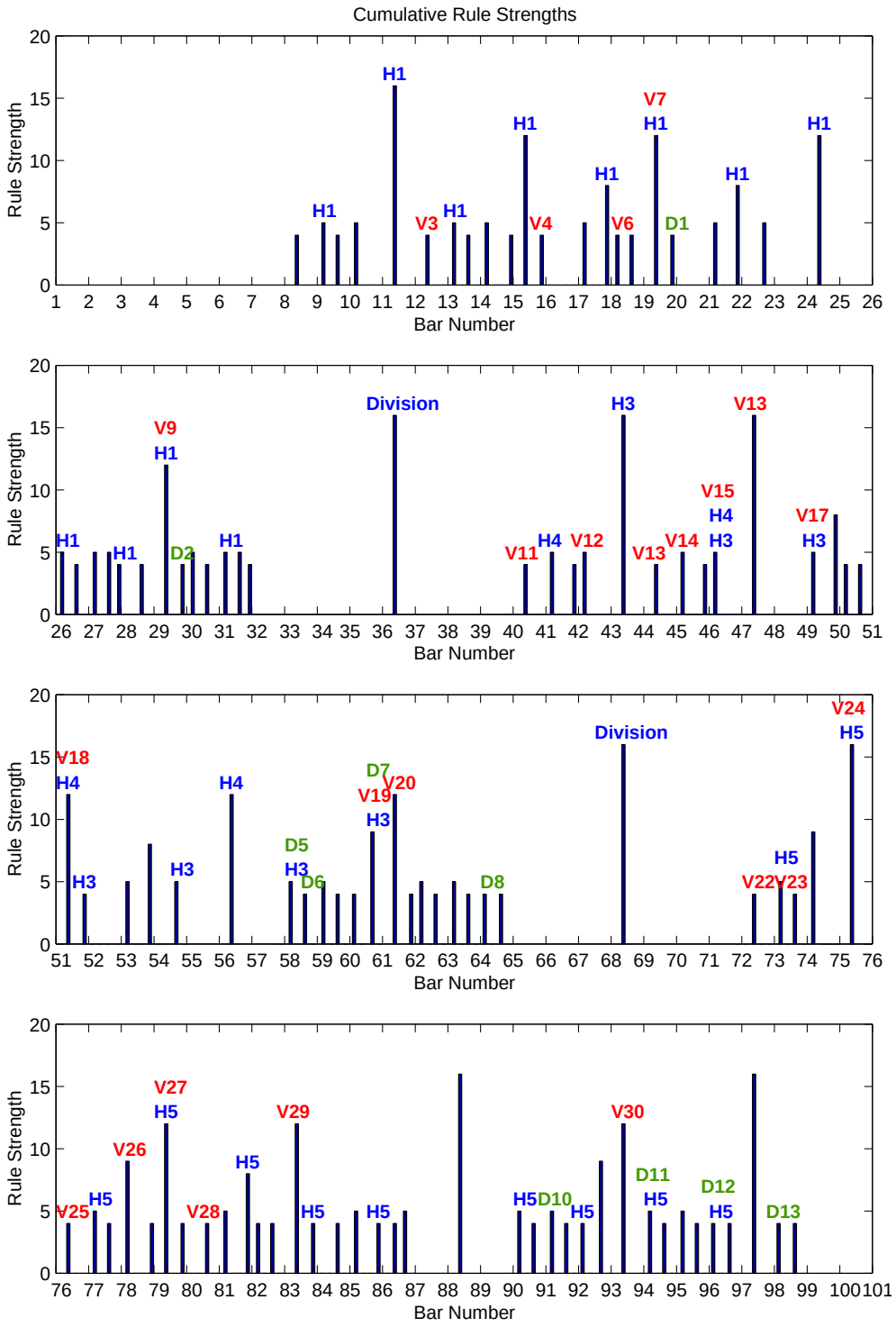


Figure 9.14: Possible grouping boundaries for *Gute Nacht*

and e017) and furthermore assigns very similar lower-level structures to each of these phrases. Interestingly, although the grouping boundaries after events e006 and e022 do correspond to the musical surface, they conflict with the lyrics of the piece which would imply boundaries after e008 and e024.

The central part of this first section again consists of a pair of motifs *bb*. As before, the synthesis process has correctly created a structure that identifies this central part as having two phrases and again assigning very similar lower-level structures to both. The same problem regarding the difference implied by the musical surface and the lyrics occurs in this section too. In this case, the lyrics would suggest boundaries after events e040 and e056. The synthesis performs most weakly during the last part of this first section during which, although it detects the two motifs *cc* correctly, it fails to create similar lower level structures for the pair.

Figure 9.16 shows the results from the thresholded vanilla analysis of the piece. The mid and high-level levels of the hierarchy are identical however, at the lower level, the vanilla analysis does produce matching pairs of structures for each of the motif pairs. Again, although these structures do correspond to those implied by the music score they contrast with the lyrics.

Second Section Figure 9.17 presents the musical structure derived for the second section of the piece. As with the first section, this consists of three pairs of motifs. The first pair of motifs is correctly identified as spanning from e095 to e126 with the split after e110. The resulting structure for both these motifs is identical.

For the remaining two pairs of motifs, the higher-level structures are correctly identified (from e127–e157 and e158–e185), however there is little similarity in the lower-level structures.

Figure 9.18 shows the results from the thresholded vanilla analysis. As with the first section, the mid and high level sections of the analyses are identical with only the lower-level structures differing. The low-level structures for the vanilla analysis do reflect the musical structure and, for the most part, produce identical analyses for each of the motifs in a pair.

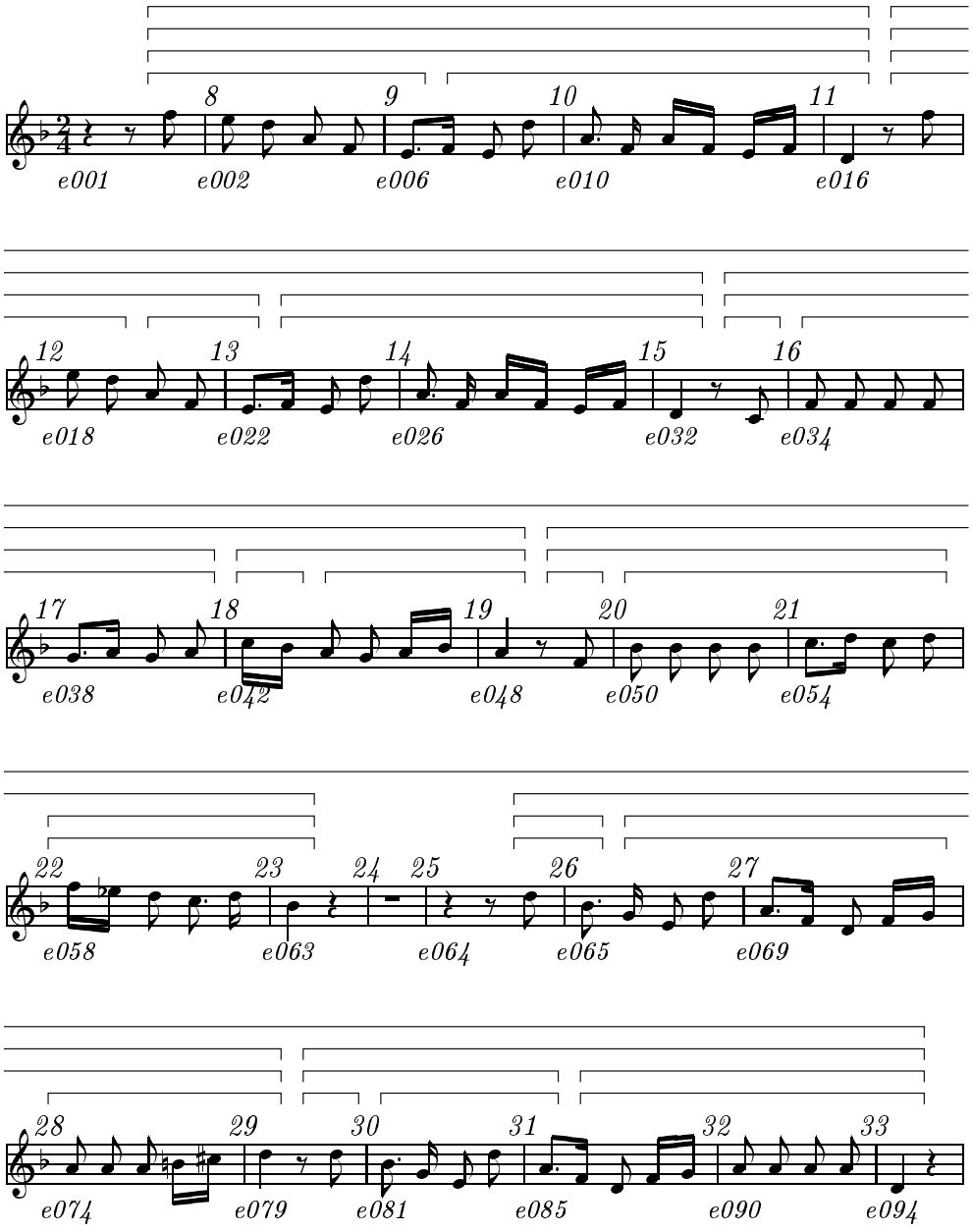


Figure 9.15: Grouping structure for the first third of *Gute Nacht* arising from the combination of the performance and structural analyses.

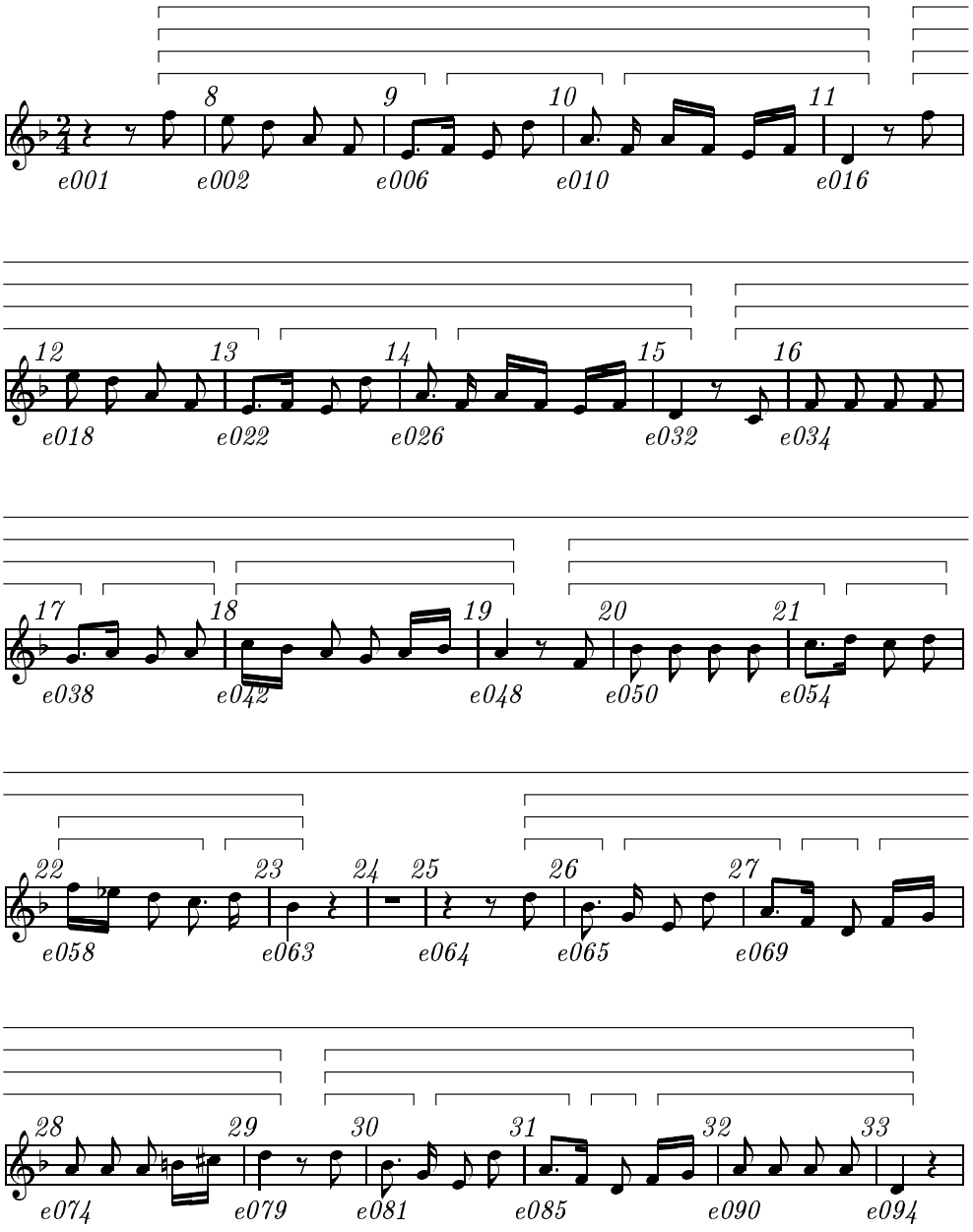


Figure 9.16: Grouping structure for the first third of *Gute Nacht* arising from the thresholded structural analysis.

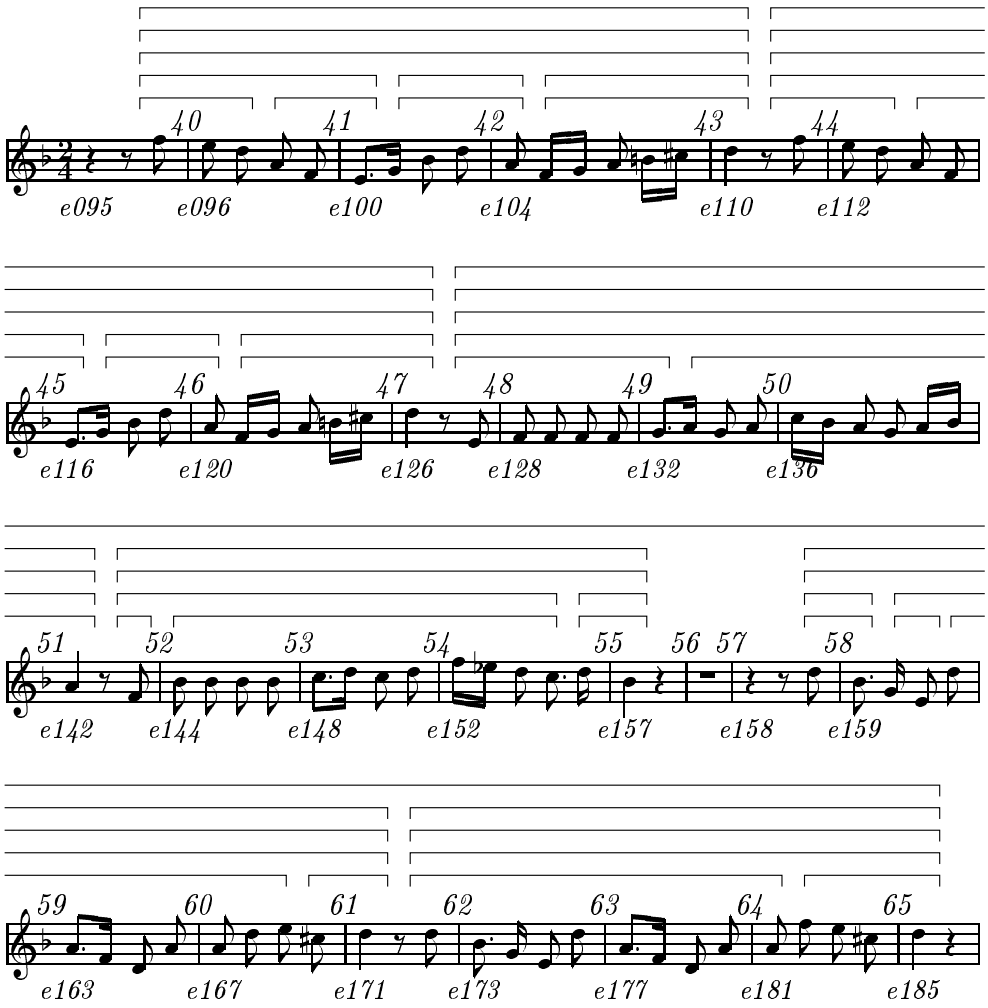


Figure 9.17: Grouping structure for the middle third of *Gute Nacht* arising from the combination of the performance and structural analyses.

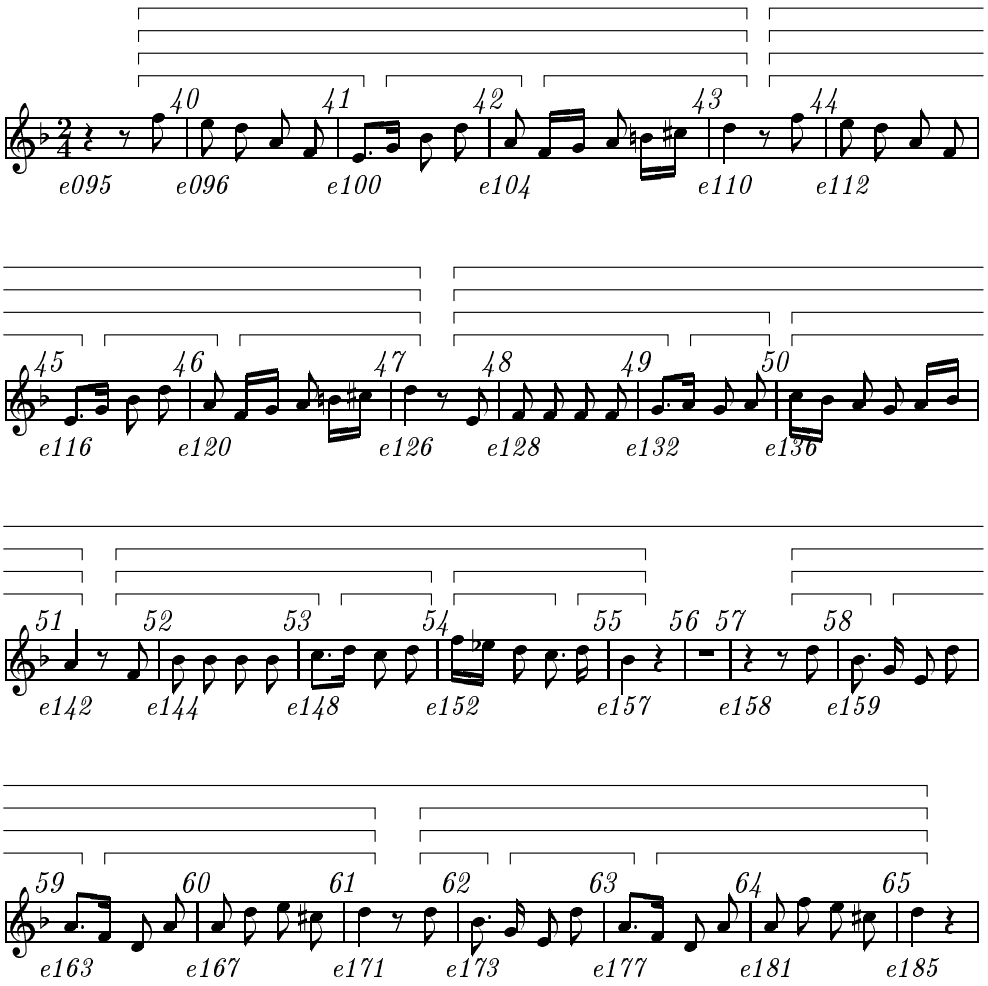


Figure 9.18: Grouping structure for the middle third of *Gute Nacht* arising from the thresholded structural analysis.

Final Section The structure for the last third of the piece is shown in Figure 9.19. In this final section of the piece, there is the common pattern of *aabbcc* which has occurred in the previous two sections. For the first pair of motifs, the analysis produces two very similar structures. However, between the second and third pair of motifs, the system fails to identify a critical boundary. This boundary, which occurs between events e249 and e250 is a fundamental feature of this section of the piece and corresponds to a relatively long rest.

However, when the performance analysis results are examined, there is not a feature which supports such a boundary, and therefore the boundary must be incorrectly marked as inactive. Apart from this serious error, the remaining structure assigned to the piece follows matches what would be expected with the remaining motif boundaries being correctly identified and with each pair being assigned similar internal structures.

The thresholded vanilla analysis, shown in Figure 9.20, is again very similar to that of the informed analysis. However, the thresholded analysis does not suffer from the critical mistake of the informed analysis and does succeed in identifying the boundary after event e249 as well as the repetition of the last two bars starting at e278.

9.5.2 Summary

The results of the synthesis process for this piece are very encouraging. The system created a detailed structure for the first third of the piece that corresponded to the musical surface. In the second and final sections, the synthesis process did not detect the low-level structure of the phrases but it did, on almost all occasions, correctly identify the higher phrase-level structure of each section. One significant mistake on the final section disrupted what would have otherwise been a very successful analysis.

As with the analysis of *Berceuse*, the fact that related phrases in the piece received similar grouping analyses supports the assertion made in Section 4.5.2 that concepts such as parallelism are implicit in the performance analysis.

Similarly to the informed analysis, the vanilla analysis produces very good mid and high-level structural analyses of the sections but does not produce the correct low-level structure. In some cases this is due to the naïve application of feature-category

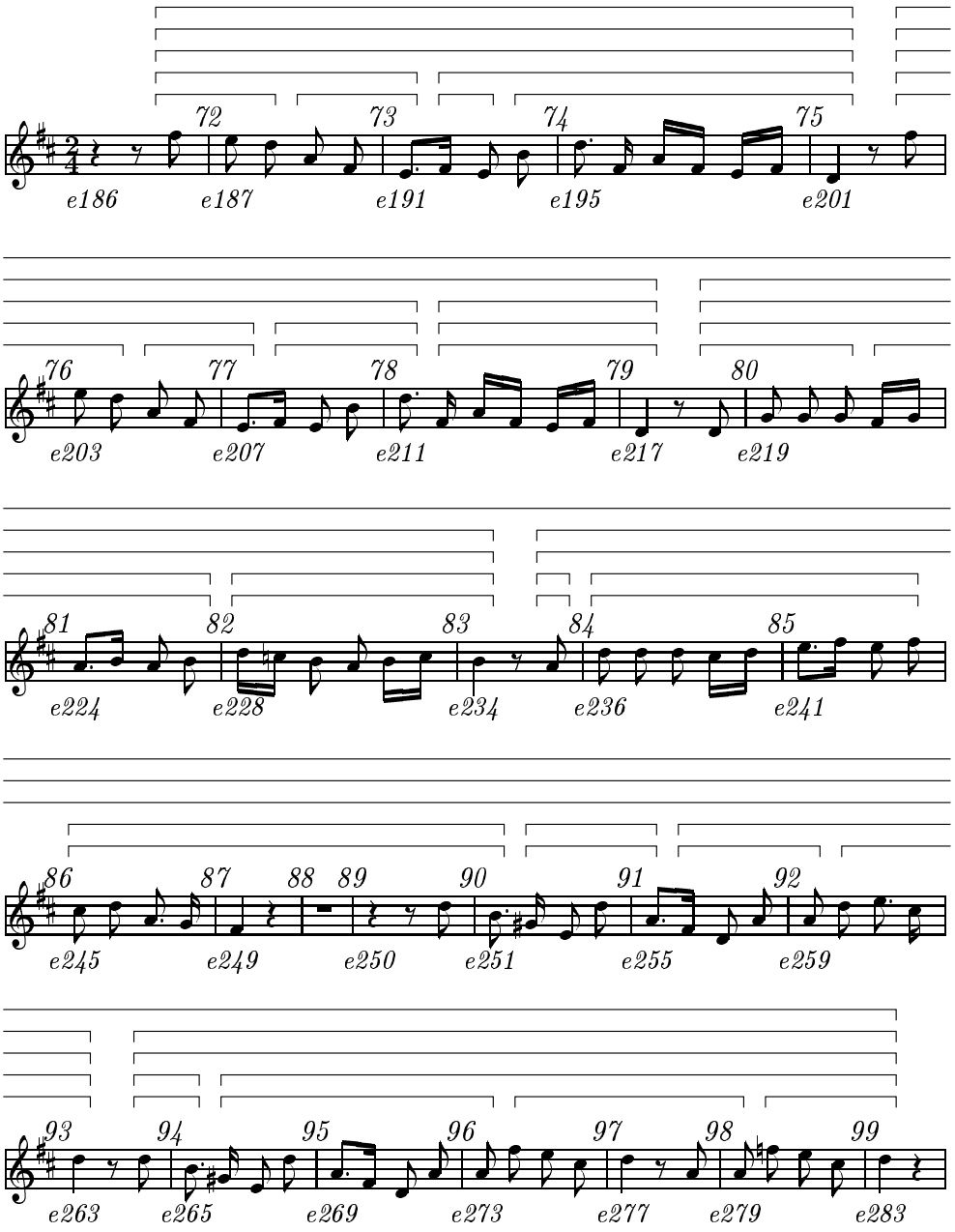


Figure 9.19: Grouping structure for the final third of *Gute Nacht* arising from the combination of the performance and structural analyses.

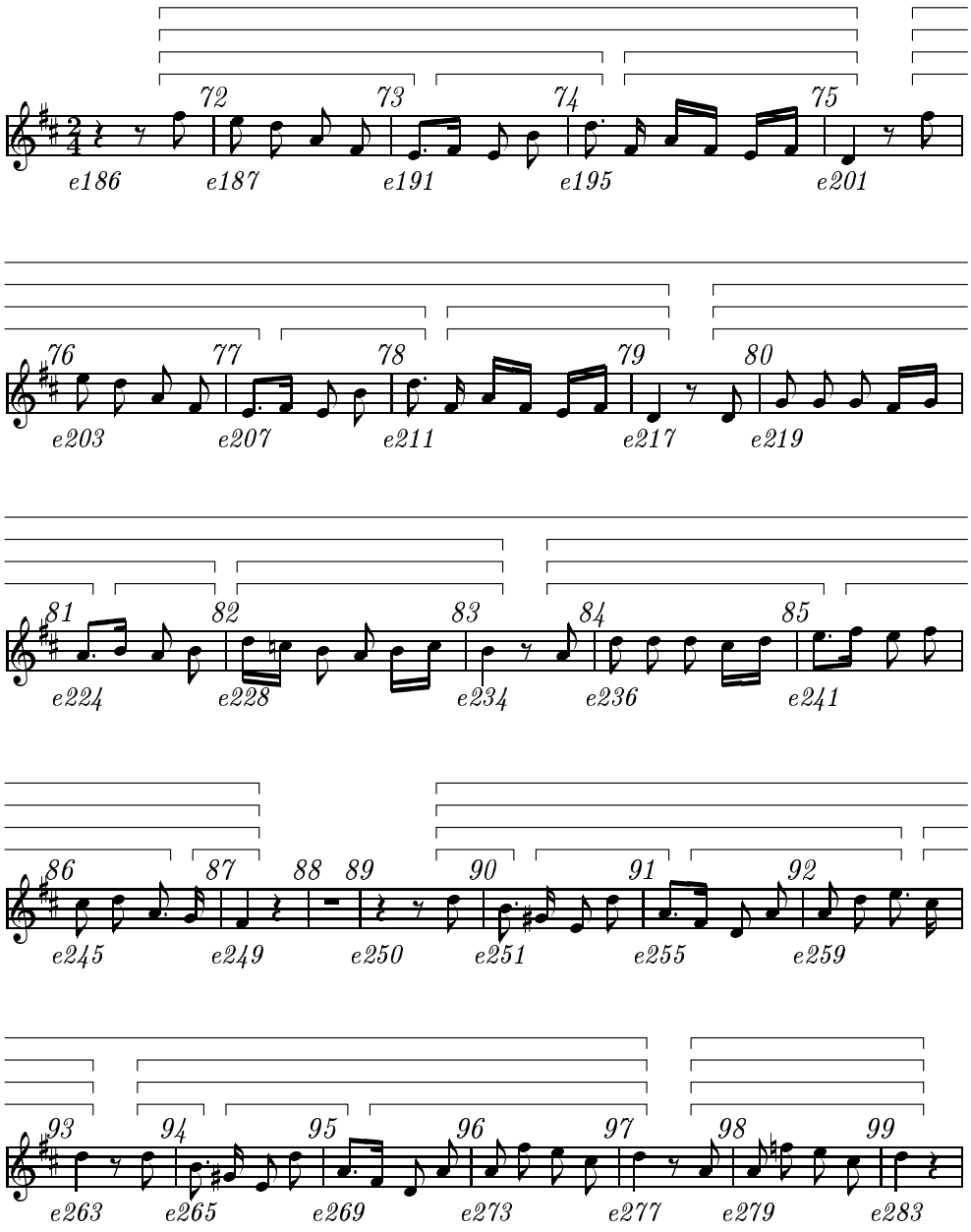


Figure 9.20: Grouping structure for the final third of *Gute Nacht* arising from the thresholded structural analysis.

weights which do not take into account the context of the application. For example, the boundary after e008 should be much more dominant than the boundary after e006 but because of the static nature of the weights, the latter of these drives the grouping structure.

9.6 Enhancements

Boundary Weights As was seen in some of the analysed pieces, the weights of the grouping analysis dictated, to a large extent, the final shape of the grouping structure. The development of a more dynamic method of weight assignment could improve the resulting structural analysis. For example, if a number of different performance features intersect at a boundary, that may suggest that that particular boundary should be assigned a greater weight that a similar boundary with only one feature confirming it. Another example would be if one performance feature is particularly pronounced relative to the other ones, then perhaps the boundary it indicates should be assigned a stronger weight.

Informed Analyses The results from one analysis could be used to contribute towards the results of another. This feedback would operate in two directions, from the grouping analysis to the performance analysis and vice-versa.

The grouping analysis could inform the performance analysis about where to look for features in the performance. For example, if the grouping analysis suggests the existence of a grouping boundary at a certain point, the performance analysis could search for sets of features which would include that point. In contrast, if the performance analysis detected a large feature that did not have a corresponding boundary in the grouping analysis, the system could search for an application of a grouping rule, e.g. parallelism, which would then generate a corresponding boundary point.

9.7 Summary

This chapter presented the synthesis process which joins the results from the structural analysis and the performance analysis together. The features detected in the performance analysis are used to either activate or deactivate potential grouping boundaries identified by the structural analysis. Once the boundaries have been marked as active, a grouping structure for the piece can be generated whose hierarchical shape is determined by the strength of the grouping boundaries.

The synthesis process was applied to the three pieces used in the empirical study with mixed results. The results for the first piece, *Berceuse*, showed that the synthesis process performed very well for the first and final sections of the piece but less so for the middle section. The resulting grouping structures for the first and final sections showed considerable agreement with the musical surface and a number of incorrect boundary points were eliminated. In the central section, where the synthesis was weakest, both analysis techniques detected similar two-bar structures but because the structures they found were misaligned, they did not support each other.

Applying the synthesis process to *Auf dem Hügel sitz ich spähend* produced the weakest results of the three pieces. Not only did the process fail to identify most of the structural features present in the piece, it managed to create false-positives on a number of occasions. Less information was gleaned from the performance analysis of this piece than from the others which is not surprising given the relatively large amount of deviation that was present across its performances (see Chapter 6).

The results for final piece, *Gute Nacht*, showed how the synthesis approach performed very well in the first two sections of the piece but less so in the last section. In the first section, the synthesis technique was able to identify both the high-level 6 phrase structure of the section and also the lower-level within phrase structures. In the later two sections, the synthesis process did not perform quite so well in detecting the lower-level within phrase structures but did, generally, capture the larger scale phrase level grouping.

The results of the synthesis process were contrasted against a thresholded vanilla analysis of the pieces. For the first piece, *Berceuse*, the informed analysis produced a structural analysis that more accurately reflected the structure of the music than just a

vanilla analysis alone. However, both analyses were generally very similar.

For the second piece, *Auf dem Hügel sitz ich spähend*, both the informed and vanilla analyses performed badly. For the final piece, *Gute Nacht*, both analyses again performed well with the vanilla analysis gaining the advantage due to the mistake in the high-level structure made by the informed analysis. The overall similarity of the results indicates that the synthesis process is capable of finding the important structural clues in the performance and using them to select important boundary features.

Overall, the synthesis process combines the information from the structural analyses and the performance analyses in a way that captures the musical structure of two of the three pieces. The binary nature of the process, which means a boundary is on if and only if there is an exactly corresponding performance feature, forces the process to fail too easily. For example, despite the fact that both analysis techniques strongly identified a two-bar repeating structure in the central section of *Berceuse*, because the analysis did not align perfectly, all that information was lost.

This chapter shows that the synthesis approach does work well for two of the chosen pieces, and, where it fails, it could be improved by the implementation of a less naïve matching process.

Chapter 10

Conclusions and Further Work

I may not have gone where I intended to go,
but I think I have ended up where I intended to be.
Douglas Adams (1952–2001)

This chapter begins by summarising the research presented in this dissertation. Some topics identified for further work are then presented. The chapter closes with the conclusions that can be drawn from this work.

10.1 Summary and Critical Analysis

This section begins by re-stating three main goals which were the focus of the research in this dissertation:

1. Provide a rule based implementation of a proven theory of musical structure that supports multiple structures and then subsequent disambiguation.

2. Develop a technique for analysing musical performance in order to provide information about the musical structure of a piece.

3. Produce a structural interpretation of a piece based upon its performance using results from the above two goals.

The work described in Chapters 4 and 5 enabled the first of these goals to be achieved. The structural analysis was based on Lerdahl and Jackendoff's (1983) Grouping Structure which, due to the nature of the preference rules, has the desired property

of supporting multiple possible structures. A new system was developed which automatically identified potential grouping boundaries in suitably encoded pieces of music.

This implementation of the grouping rules included feature-category weighting factors which were primarily based on descriptive text in GTTM. These weighting factors express the relative importance of each preference rule when compared to the others. The weighting factors influence the strength of the detected grouping boundaries which in turn affects the shape of the final grouping hierarchy.

The implementation of the grouping rules proved to be very successful in terms of identifying potential grouping boundaries.[1] The feature-category weighting factors added a useful source of information when deriving the higher-level structure of the piece. The resulting analyses were, on the whole, representative of the structure of the pieces and examples studied.

The main weakness of the grouping analysis component is that it does not take into account the context of the rules when assigning a boundary strength. Currently the component simply calculates the total weight of the rules that apply at that point, it uses no information from the context in which those rules applied. For example, the component does not draw any distinction between a quarter-note rest and a rest that lasts several bars. It is highly likely that the second of these situations should create a boundary of greater strength than the first.

The grouping boundaries detected by this component were then applied to a novel tree representation described in Chapter 5. The representation provided an effective means of representing the grouping structure as a tree-based structure which supports the dynamic introduction of separating boundaries between nodes.

Each musical event was represented by a leaf-node of the tree and the potential grouping boundaries were represented as constraints that, when active, forced the nodes to belong to different branches. The height of the separation was dictated by the strength of the boundary at that point when compared with neighbouring boundaries. Each of these boundaries has a switch to indicate whether it should be considered active or not. If a boundary was not active, the leaf nodes either side of it were joined together onto the same branch.

[1] It even succeeded in finding some errors in the examples presented by Lerdahl and Jackendoff.

An empirical study was then performed in order to gather some performance data to work with (Chapter 6). The study had two main aims: the first was to show that performers tend to play the same piece the same way over a series of performances; the second aim was to gather performance data for analysis. The results of the study showed that performers did tend to perform the piece in a consistent way, corroborating Repp's (1996) results, and as a concrete output we obtained fifteen performances to work with during the rest of this research.

The performance data was then subjected to a novel analysis technique to try to detect significant structural features from the performance (the second goal). The analysis technique produced some mixed results (Chapter 7 and Chapter 8). For two of the pieces, *Berceuse* and *Gute Nacht*, a large set of potential features were discovered which gave indications of phrase starting and ending points and also of repetitive phrase structures. For the other piece, *Auf dem Hügel sitz ich spähend*, the resulting data from the analysis only weakly indicated a few features.

The results from the analyses were a set of features found in the performances intended to indicate significant structural boundaries. These sets of features were then passed to the synthesis module (Chapter 9). The features were used to select which boundaries from the grouping analysis should be set to active (the final goal).

For two of the pieces, *Berceuse* and *Gute Nacht*, the results are very encouraging. For the other piece, *Auf dem Hügel sitz ich spähend*, the results were disappointing. In each case, the technique did manage to uncover some significant boundaries and generally performed better on detecting the mid-level and higher-level structural features rather than the low-level ones.

10.2 Further Work

The nature of this thesis means that it touches upon a broad range of topics. Because of this there are a large number of avenues that deserve further investigation. The following sections, one for each of the core chapters, contain a brief summary of potential future work which reinforce some of the ideas presented in this thesis or generate new topics for research.

10.2.1 Structural Analysis

The current implementation of Lerdahl and Jackendoff's grouping structure is not complete; it does not include transformation rules or parallelism. As was mentioned in Chapter 4, parallelism is the least defined yet potentially most powerful of the grouping preference rules so it would be interesting to see the effect of applying parallelism to the existing structural analyses.

The structural analysis component consists of an implementation of the Grouping Rules, the other three aspects of GTTM (which are also hierarchical) could be added. The addition of these extra components would help reduce some of the less significant boundaries and lend weight to some of the more significant ones.

The feature-category weights chosen for the grouping rules were selected based on very limited experimentation to suit the three pieces being analysed. These are very important in terms of deciding the overall shape of the hierarchy, so it would be desirable to have more experimental results regarding these weights.

Finally, the structural component should work with any theory of musical structure which is hierarchically based, so it would be interesting to see some experimentation into the suitability of other structural theories, such as GCTMS, in the current system.

10.2.2 Tree Representation

The current representation is very computationally intensive and needs further optimising. Specifically a better way of implementing the repel function which, necessarily, causes the delay of the majority of the computation. The constraint system also supports the introduction of user defined algorithms to solve the constraints, the potential for more efficient and specialised algorithm exists.

The principle behind the representation seems to be an interesting concept, it would be worthwhile investigating the effects of different underlying representations that maintain the same overall idea.

10.2.3 Empirical Study

As with most experiments, it would be desirable to have collected more data from more subjects. More specifically it would be interesting to compare how different performers play the same pieces and how that affects the performance analysis phase. In order to increase the confidence in the techniques presented here, it would also be desirable to gather a larger set of performances of more pieces by different performers.

Also, the breaks intended to simulate a break between rehearsals did not have a significant effects on the performances. A further study which introduced longer breaks of days or weeks between performances would provide more information on the rôle that time plays on how a musician performs a piece.

10.2.4 Performance Analysis

The novel use of an interpolation technique to complete the performance data set performed well for the majority of the performances. However, in some cases the interpolated results clearly differed from ones which were musically sensible. An enhanced interpolation method is needed which can take account of the musical nature of these events and is capable of correcting itself when non-musical events (such as those with negative durations) are produced.

The performance analysis technique did work well. The data generated by the performance analysis was sufficiently detailed to produce results for the feature detection process. Some refinement of the process could be conducted to investigate issues such as the error penalty applied to concave curves.

The analysis technique used in this thesis relies on a the music under analysis having a regular phrase structure. This constraint limits the number of pieces which can be analysed using this technique. A different analysis technique based on phrase-final lengthening could be used, however the task of separating false-positives from the analysis results will be more difficult without the supporting evidence of surrounding phrases which is obtained by the approach used in this thesis.

Applying more analysis techniques may provide a way of detecting other features that the phrase-final lengthening based model did not detect. For example, low-level

details are not detected well by the current implementation. The interaction of a set of analysis techniques would also be worth investigating: the results of different analyses could be joined to positively reinforce one another; analyses could be guided with results from other ones (i.e. if one analysis detected a particular feature, evidence for that feature in other analyses would be actively sought).

10.2.5 Feature Detection

This process needs to be automated. The current set of features were obtained by visual inspection of the graph which introduces the possibility of bias and errors. Ideally this stage of the process would be fully automated; a feasible challenge given the rigorous specifications of the features which are being detected.

10.2.6 Synthesis

The synthesis stage was also performed manually for this research. The current synthesis process only acts to confirm or reject boundaries, however it is obvious from the performance data that some of the features found by the performance analysis are more significant than others. If the significance of the detected boundaries could be measured and applied in conjunction with the feature-category weights then the resulting hierarchy would reflect more closely the performance data.[2]

Also, the matching of features to structural boundaries offers another interesting challenge. It can be non-trivial to decide which boundary is indicated by which feature. Heuristics such as preferring the boundary with the greater feature-category weight may be of some assistance when automating this process.

10.3 Conclusions

The indications of this research suggest that the idea has merit. As discussed above, for the first piece, *Berceuse*, the technique as a whole performed extremely well. The

[2] It is currently dominated by the values of the feature-category weights.

combination of grouping boundaries, weighting factors and performance analysis produced a result that was almost perfect for the first and final sections of the piece. The system performed least well during the central section of the piece where the structure was least imposing.

For the second piece, *Auf dem Hügel sitz ich spähend*, the results of the structural analysis seemed plausible however the performance analysis provided few features to work with. When these few features were applied to the structural analysis, they rarely coincided, and when they did, the results were disappointing. The system failed to produce an accurate structural analysis for this piece.

For the final piece, *Gute Nacht*, the results of the structural analysis were again plausible and in this case, the majority of features identified by the performance analysis did correspond to important structural features. The combination of the two analyses produced a musical structure that was accurate for a significant majority of high-level features and succeeded in identifying some low-level ones.

Overall the results are promising, with some very positive results for two of three pieces. However, the research presented in this thesis is necessarily broad which means that some of the facets of the work, such as the feature-category weighting factors, were not explored as fully as desired.

This thesis has presented a novel idea aimed at creating a computational system that can take a more active rôle when accompanying a human musician. The idea is intuitive and the results indicate that the idea has potential and with some further refinement may prove very successful.

An appealing side effect of the research in this thesis is the large scope for further work which has become apparent. The design of the system will allow it to be used as a test-bed to test theories of structural analysis, performance analysis and expressive performance generation. The results of this further work, along with the other ideas mentioned above, would offer some significant contributions towards the fields of Musical Performance Modelling and Artificial Intelligence.

List of Acronyms

Charm . . . . . Common Hierarchical Abstract Representation for Music An abstract data type which allows the specification of music as events, constituents and associated operations (Smaill et al., 1993).

GCTMS . . . . . General Cognitive Theory of Musical Structure A theory of musical structure proposed in Cambouropoulos (1998). The theory is based on cognitive principles and is intended to be independent of style or idiom.

GPR . . . . . Grouping Preference Rule A rule from GTTM which is used to select from the set of all grouping structures the one that most closely resembles the musical surface.

GTTM . . . . . Generative Theory of Tonal Music A formal theory of musical structure proposed by Lerdahl and Jackendoff (1983).

GWFR . . . . . Grouping Well-Formedness Rule A rule from GTTM which specifies the shape of the grouping hierarchy without specialising it to a particular piece.

IOI . . . . . Inter-Onset Interval The time period which occurs between the onsets of two musical events.

IRM . . . . . Implication-Realisation Model A theory of musical structure based upon the ideas of expectation and realization (Narmour, 1992). The theory consists of two processes: top-down processes which are id-

iom specific and bottom-up processes which are based on Gestalt principles.

LBDM ..... Local Boundary Detection Model A model from GCTMS which attempts to identify local boundaries in a musical surface.

MIDI ..... Musical Instrument Digital Interface A standard used to enable different electronic musical devices, such as keyboards and sound-cards to communicate with each other.

SPMA ..... Sequential Pattern Matching Algorithm A pattern induction algorithm, used by GCTMS which takes a musical surface and discovers all possible patterns which are implied by the surface (including overlapping patterns).

Glossary

Accelerando An indication that a sequence of events should be performed increasingly quickly. *(p 13)*

Accent An emphasis of a particular event, usually achieved through the application of changes in dynamics or timing. *(p 16)*

Amplitude Envelope The curve that represents the loudness of a musical event during the lifetime of that event. *(p 8)*

Arpeggio The performance of the notes of a chord in sequence rather than in parallel. *(p 24)*

Articulation The way in which an event is performed, usually controlled by altering the attack or decay. *(p 12)*

Attack The shape of the amplitude envelope at the start of a musical event. *(p 12)*

Crescendo An indication over a series of events which requests that, as the musician performs the events, the later ones are louder in comparison to the earlier ones. *(p 14)*

Decrescendo An indicator to the performer to play a series of notes in such a way that they gradually decrease in loudness. *(p 14)*

Dynamics The aspect of performance concerned with the loudness of events. *(p 7)*

Expressive Deviation A deliberate manipulation of the timing or dynamics of a musical event in order to emphasise an aspect of the musical structure. *(p 9)*

Expressive Performance A performance of a piece of music that makes use of timing, dynamics, timbre, etc. to highlight features of the piece. *(p 1)*

Harmonic Progression A description of how the harmonic structure of a piece changes over its duration. *(p 16)*

Intervallic The distance between two musical pitches. *(p 45)*

Legato A notation requesting that the effects are played smoothly. *(p 12)*

Lento ed Espressivo An indication that the piece should be played at a slow tempo and expressively. *(p 226)*

Mechanical Performance A performance of a piece of music that rigidly obeys the timing and other annotations of the score. *(p 7)*

Motif A musical idea used as a building block for a theme. *(p 16)*

Musical Score A symbolic representation of a piece of music. The musical score is a very rich representation and includes references to pitch, timing, articulation, etc. *(p 7)*

Musical Structure A broad term covering many different structural aspects including, but not limited to, metrical structure, harmonic progression, phrasal structure, etc. *(p 1)*

Musical Surface A piece of music as viewed on a note-to-note basis. *(p 44)*

Ornament An embellishing note not belonging to the essential harmony or melody. *(p 23)*

Phrase Structure A piece of music is often heard as a series of motifs, phrases and sections. Phrases form the intermediate part of a musical structure. *(p 14)*

Phrase-Final Lengthening The tendency of human musicians to lengthen musical events which occur towards the end of a phrase. *(p 137)*

Piano A score notation indicating that the affected notes should be performed softly. *(p 100)*

Pitch The position of an event on a musical scale. *(p 16)*

Rallentando A score notation indicating that the affected notes should be performed increasingly slowly. *(p 100)*

Ritardando See Rallentando. *(p 13)*

Rubato The changing timing of a section of music with respect to a steady beat. *(p 14)*

Sempre Dolce An annotation requesting that the section is to be performed softly, literally “always sweetly”. *(p 100)*

Staccato An articulation that requests that every note is played distinctly from the surrounding notes. *(p 12)*

Swell An increase in loudness which is then followed by a subsequent decrease. *(p 97)*

Syncopation A shifting of the rhythmic accent which means that a normally unaccented beat adopts a more significant rôle. *(p 164)*

Tactus The underlying beat of a piece of music. *(p 20)*

Tempo The speed at which a musical piece is performed. *(p 8)*

Tension The feeling of expectation created when, for example, a piece of music deviates from an established repetitive structure. *(p 16)*

Theme A melody on which a piece, or section of a piece, is based. *(p 16)*

Timbre The “quality” of a sound. *(p 9)*

Timing The aspect of performance concerned with the properties of events with respect to time such as start-time, duration, etc. *(p 7)*

Vibrato The repeating fluctuation in pitch of a performed event. *(p 12)*

Bibliography

Lucy J. Appleton, W. Luke Windsor, and Eric F. Clarke. Cooperation in piano duet performance. In *Proceedings of the European Society for the Cognitive sciences Of Music Conference*, pages 471–474, 1997.

Josep Lluís Arcos, Ramon López de Mántaras, and Xavier Serra. Saxex: a case-based reasoning system for generating expressive musical performances. In *Proceedings of the International Computer Music Conference*, pages 329–336, 1997.

G. E. P. Box and G. M. Jenkins. *Time Series Analysis: Forecasting and control*. San Francisco: Holden-Day, 1976.

E. Cambouropoulos, T. Crawford, and C.S. Iliopoulos. Pattern processing in melodic sequences: Challenges, caveats and prospects. In *Proceedings of the AISB'99 Convention (Artificial Intelligence and Simulation of Behaviour)*, 1999.

Emilios Cambouropoulos. *Towards a General Theory of Musical Structure*. PhD thesis, Faculty of Music, University of Edinburgh, 1998.

Sergio Canazza, Giovanni De Poli, Antonio Rodà, and Avlise Vidolin. Analysis by synthesis of the expressive intentions in musical performance. In *Proceedings of the International Computer Music Conference*, pages 113–120, 1997.

Sergio Canazza and Antonio Rodà. Adding expressiveness in musical performance in real time. In *The Society for the Study of Artificial Intelligence and the Simulation of Behaviour: Symposium on Musical Creativity*, pages 134–139, April 1999.

Mats Carlsson, Greger Ottosson, and B. Carlson. An open-ended finite domain constraint solver. In *Proceedings of Programming Languages: Implementations, Logics and Programs*, 1997.

Arther Cayley. On the analytical forms called trees. In *Collected Mathematical Papers*, volume 4. Cambridge University Press, 1891.

Eric F. Clarke. Imitating and evaluating real and transformed musical performances. *Music Perception*, 10(3):317–341, 1993.

Eric F. Clarke and Carol L. Krumhansl. Perceiving musical time. *Music Perception*, 7 (3):213–252, Spring 1990.

Ben Curry and Geraint A. Wiggins. A new approach to cooperative performance: A preliminary experiment. *International Journal of Computing Anticipatory Systems*, 4:163–178, 1999.

Ben Curry, Geraint A. Wiggins, and Gillian Hayes. Representing trees with constraints. In J. Lloyd et al, editor, *Proceedings of the First International Conference on Computational Logic*, volume 1861 of *LNAI*, pages 315–325. Springer Verlag, 2000.

Roger B. Dannenberg. Music understanding by computer. In *IAKTA/LIST International Workshop on Knowledge Technology in the Arts Proceedings*, pages 41–56, 1993.

Roger B. Dannenberg and Hirofumi Mukaino. New techniques for enhanced quality of computer accompaniment. In *Proceedings of the International Computer Music Conference*, pages 243–249, 1988.

Giovanni De Poli, Antonio Rodà, and Alvise Vidolin. Note-by-note analysis of the influence of expressive intentions and musical structure in violin performance. *Journal of New Musical Research*, 27(3), 1998.

Irène Deliège. Grouping conditions in listening to music: An approach to Lerdahl and Jackendoff's grouping preference rules. *Music Perception*, 4(4):325–360, 1987.

Peter Desain and Henkjan Honing. *Music, Mind and Machine, Studies in Computer Music, Music Cognition and Artificial Intelligence*, chapter Tempo curves considered harmful. Amsterdam: Thesis Publishers, 1992.

Peter Desain and Henkjan Honing. Does expressive timing in music performance scale proportionally with tempo? *Psychological Research*, 56:285–292, 1994.

Peter Desain, Henkjan Honing, and Hank Heijink. Robust score-performance matching: Taking advantage of structural information. In *Proceedings of the International Computer Music Conference*, pages 337–340, 1997.

Peter Desain and Siebe De Vos. Autocorrelation and the study of musical expression. In *Proceedings of the International Computer Music Conference*, 1990.

Christopher Dobrain. *Max 3.5 Reference Manual*. Opcode Systems, Inc., 1999.

Anders Friberg and Roberto Bresin. Automatic musical punctuation: A rule system, and a neural network approach. In *Proceedings of KANSEI - The Technology of Emotion AIMI International Workshop*, pages 159–163, 1997.

Anders Friberg, Lars Frydén, and Johan Sundberg. A rule for automatic musical punctuation of melodies. In *Proceedings of the European Society for the Cognitive sciences Of Music Conference*, pages 719–723, 1997.

Andy Green, Chris Cannam, and Guillaume Laurent. *Rosegarden User Manual*. Published electronically at `http://www.all-day-breakfast.com/rosegarden` (April 28, 2002), 2002.

Lorin Grubb and Roger B. Dannenberg. A stochastic method of tracking a vocal performer. In *Proceedings of the International Computer Music Conference*, pages 301–308, 1997.

P. Van Henternryck. *Constraint Satisfaction in Logic Programming*. Logic Programming Series. MIT Press, 1989.

Martin Henz, Stefan Lauer, and Detlev Zimmermann. COMPOzE — intention-based music composition through constraint programming. In *Proceedings of the 8th*

IEEE International Conference on Tools with Artificial Intelligence, pages 118–121, Toulouse, France, November16–19 1996. IEEE Computer Society Press.

Perry R. Hinton. *Statistics Explained*. Routledge, 2001.

Patrik N. Juslin. Perceived emotional expression in synthesized performances of a short melody: Capturing the listener's judgement policy. *Musicae Scientiae*, 1(2): 225–256, Fall 1997.

Carol L. Krumhansl. Music psychology and music theory: Problems and prospects. *Music Theory Spectrum*, 17:53–90, 1995.

Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, 1983.

G.A. Miller. *Psychology: The science of mental life*. New York: Pelican Books, 1962.

Eugene Narmour. *The analysis and cognition of melodic complexity: the implication-realization model*. University of Chicago Press, 1992.

Richard Parncutt. Modeling piano performance: Physics and cognition of a virtual pianist. In *Proceedings of the International Computer Music Conference*, pages 15–18, 1997.

Amandine Penel and Carolyn Drake. Timing variations in music performance: hierarchical segmentation organisation and rhythmic grouping. In *Proceedings of the European Society for the Cognitive sciences Of Music Conference*, pages 465–470, 1997.

Christopher Raphael. Synthesizing musical accompaniments with bayesian belief networks. *Journal of New Music Research*, 30(1):59–67, 2001.

Bruno H. Repp. Diversity and commonality in music performance: An analysis of timing microstructure in Schumann's "Träumerei". *Journal of the Acoustical Society of America*, 92(5):2546–2568, November 1992a.

Bruno H. Repp. Probing the cognitive representation of musical time: Structural constraints on the perception of timing perturbations. *Cognition*, 44:241–281, February 1992b.

Bruno H. Repp. On determining the basic tempo of an expressive music performance. *Psychology of Music*, 22:157–167, 1994a.

Bruno H. Repp. Relational invariance of expressive microstructure across global tempo changes in music performance: an exploratory study. *Psychological Research*, 56: 269–284, 1994b.

Bruno H. Repp. Expressive timing in Schumann's "Träumerei": An analysis of performances by graduate student pianists. *Journal of the Acoustical Society of America*, 98(5):2413–2427, November 1995.

Bruno H. Repp. The dynamics of expressive piano performance: Schumann's "Träumerei" revisited. *Journal of the Acoustical Society of America*, 100(1):641–650, July 1996.

Bruno H. Repp. The aesthetic quality of a quantitatively average music performance: two preliminary experiments. *Music Perception*, 14(4):419–444, Summer 1997a.

Bruno H. Repp. Expressive timing in a Debussy prelude: A comparison of student and expert pianists. *Musicae Scientiae*, 1(2):257–268, Fall 1997b.

Carron Robbie. Implementing a generative grammar for music. Master's thesis, Department of Artificial Intelligence, University of Edinburgh, 1994.

Antonio Rodà and Sergio Canazza. Adding expressiveness in musical performance in real time. In *International Conference on Multimedia Computing and Systems*, volume 2, pages 1026–1027, 1999.

J. Rothstein. *MIDI: A Comprehensive Introduction*. Oxford University Press, 1992.

N. J. A. Sloane. *The On-line encyclopedia of integer sequences*. Published electronically at http://www.research.att.com/˜njas/sequences/, 2000.

Alan Smaill, Geraint Wiggins, and Mitch Harris. Hierarchical music representation for composition and analysis. *Computers and the Humanities*, 27:7–17, 1993.

Dale R. Stammen and Bruce Pennycook. Real-time segmentation of music using an adaption of lerdahl and jackendoff's grouping principles. In *Proceedings of the International Conference on Music Perception and Cognition*, 1994.

Neil Todd. A model of expressive timing in tonal music. *Music Perception*, 3(1): 33–58, Fall 1985.

Neil Todd. A computational model of rubato. *Contemporary Music Review*, 3:69–88, 1989a.

Neil Todd. Towards a cognitive theory of expression: The performance and perception of rubato. *Contemporary Music Review*, 4:405–416, 1989b.

Neil Todd. The dynamics of dynamics: A model of musical expression. *Accoustical Society of America*, 91(6):3540–3550, June 1992.

Neil Philip Todd. *Computation Theory and Implementations of an Abstract Expression System: A Contribution to Computation Psychomusicology*. PhD thesis, Faculty of Science, University of Exeter, 1989c.

Kensei Tsuchida, Yoshihiro Adachi, Takanori Imaki, and Takeo Yaku. Tree drawing using constraint logic programming. In *International Conference on Logic Programming*, 1997.

Gerhard Widmer. A machine learning analysis of expressive timing in pianists' performances of Schumann's "Träumerei". In *Proceedings of the Stockholm Symposium on Generative Grammars for Music Performance*, 1995a.

Gerhard Widmer. Modeling the rational basis of musical expression. *Computer Music Journal*, 19(2):76–96, 1995b.

Gerhard Widmer. Learning expressive performance: The structure-level approach. *Journal of New Music Research*, 25(2), 1996.

Gerhard Widmer. Applications of machine learning to music research: Empirical investigations into the phenomenon of musical expression. In I.Bratko R.S. Michalsku and M.Kubat, editors, *Machine Learning and Data Mining: Methods and Applications*. John Wiley & Sons Ltd, 1997.

Gerhard Widmer. On the potential of machine learning for music research. In E. Miranda, editor, *Readings in Music and Artificial Intelligence*. London: Harwood Academic Publishers, 2000.

Geraint Wiggins, Mitch Harris, and Alan Smaill. Representing music for analysis and composition. In M. Balaban, K. Ebcioglu, O. Laske, C. Lischka, and L. Sorisio, editors, *Proceedings of 2nd IJCAI AI/Music Workshop*, pages 63–71, 1993.

Appendix A

Charm Representation

A.1 Fauré's *Berceuse*

```
piece(faure, [
        event(e001, [b, =, 6], 008, 2, []),
        event(e002, [b, =, 6], 010, 3, []),
        event(e003, [g, #, 6], 013, 1, []),
        event(e004, [a, =, 6], 014, 1, []),
        event(e005, [b, =, 6], 015, 1, []),
        event(e006, [g, #, 6], 016, 1, []),
        event(e007, [e, =, 6], 017, 1, []),
        event(e008, [f, #, 6], 018, 1, []),
        event(e009, [g, #, 6], 019, 1, []),
        event(e010, [e, =, 6], 020, 2, []),
        event(e011, [b, =, 5], 022, 1, []),
        event(e012, [b, =, 6], 024, 2, []),
        event(e013, [b, =, 6], 026, 3, []),
        event(e014, [g, #, 6], 029, 1, []),
        event(e015, [a, =, 6], 030, 1, []),
        event(e016, [b, =, 6], 031, 1, []),
        event(e017, [g, #, 6], 032, 1, []),
        event(e018, [e, =, 6], 033, 1, []),
        event(e019, [e, =, 7], 034, 1, []),
        event(e020, [d, #, 7], 035, 1, []),
        event(e021, [c, #, 7], 036, 2, []),
        event(e022, [b, =, 6], 038, 1, []),
        event(e023, [f, #, 7], 040, 2, []),
        event(e024, [f, #, 7], 042, 3, []),
        event(e025, [d, #, 7], 045, 1, []),
```

```
event(e026, [e, =, 7], 046, 1, []),
event(e027, [f, #, 7], 047, 1, []),
event(e028, [d, #, 7], 048, 1, []),
event(e029, [b, =, 6], 049, 1, []),
event(e030, [c, #, 7], 050, 1, []),
event(e031, [d, #, 7], 051, 1, []),
event(e032, [b, =, 6], 052, 2, []),
event(e033, [f, #, 6], 054, 1, []),
event(e034, [f, #, 7], 056, 2, []),
event(e035, [f, #, 7], 058, 3, []),
event(e036, [g, #, 7], 061, 1, []),
event(e037, [e, =, 7], 062, 1, []),
event(e038, [c, #, 7], 063, 1, []),
event(e039, [e, =, 7], 064, 1, []),
event(e040, [f, #, 7], 065, 1, []),
event(e041, [d, #, 7], 066, 1, []),
event(e042, [b, =, 6], 067, 1, []),
event(e043, [d, #, 7], 068, 2, []),
event(e044, [c, #, 7], 070, 2, []),
event(e045, [b, =, 6], 072, 2, []),
event(e046, [b, =, 6], 074, 3, []),
event(e047, [g, #, 6], 077, 1, []),
event(e048, [a, =, 6], 078, 1, []),
event(e049, [b, =, 6], 079, 1, []),
event(e050, [g, #, 6], 080, 1, []),
event(e051, [e, =, 6], 081, 1, []),
event(e052, [f, #, 6], 082, 1, []),
event(e053, [g, #, 6], 083, 1, []),
event(e054, [e, =, 6], 084, 2, []),
event(e055, [b, =, 5], 086, 1, []),
event(e056, [b, =, 6], 088, 2, []),
event(e057, [b, =, 6], 090, 3, []),
event(e058, [g, #, 6], 093, 1, []),
event(e059, [a, =, 6], 094, 1, []),
event(e060, [b, =, 6], 095, 1, []),
event(e061, [g, #, 6], 096, 1, []),
event(e062, [e, =, 6], 097, 1, []),
event(e063, [e, =, 7], 098, 1, []),
event(e064, [d, #, 7], 099, 1, []),
event(e065, [c, #, 7], 100, 2, []),
event(e066, [b, =, 6], 102, 1, []),
event(e067, [e, =, 7], 104, 2, []),
event(e068, [e, =, 7], 106, 3, []),
event(e069, [c, #, 7], 109, 1, []),
```

```
event(e070, [b, =, 6], 110, 1, []),
event(e071, [a, =, 6], 111, 1, []),
event(e072, [e, =, 7], 112, 2, []),
event(e073, [e, =, 7], 114, 3, []),
event(e074, [c, #, 7], 117, 1, []),
event(e075, [b, =, 6], 118, 1, []),
event(e076, [a, =, 6], 119, 1, []),
event(e077, [b, =, 6], 120, 2, []),
event(e078, [b, =, 7], 122, 3, []),
event(e079, [g, #, 7], 125, 1, []),
event(e080, [f, #, 7], 126, 1, []),
event(e081, [e, =, 7], 127, 1, []),
event(e082, [g, #, 7], 128, 2, []),
event(e083, [f, #, 7], 130, 2, []),
event(e084, [e, =, 7], 132, 3, []),
event(e085, [e, =, 6], 136, 2, []),
event(e086, [e, =, 6], 138, 3, []),
event(e087, [c, =, 6], 141, 1, []),
event(e088, [d, =, 6], 142, 1, []),
event(e089, [e, =, 6], 143, 1, []),
event(e090, [e, =, 6], 144, 2, []),
event(e091, [c, =, 6], 146, 3, []),
event(e092, [g, =, 5], 149, 1, []),
event(e093, [a, =, 5], 150, 1, []),
event(e094, [c, =, 6], 151, 1, []),
event(e095, [b, =, 5], 152, 1, []),
event(e096, [d, =, 6], 153, 1, []),
event(e097, [g, =, 6], 154, 3, []),
event(e098, [e, =, 7], 157, 1, []),
event(e099, [g, =, 7], 158, 3, []),
event(e100, [a, #, 5], 161, 1, []),
event(e101, [d, =, 6], 162, 1, []),
event(e102, [g, =, 6], 163, 1, []),
event(e103, [f, =, 6], 164, 1, []),
event(e104, [e, =, 6], 165, 1, []),
event(e105, [d, =, 6], 166, 1, []),
event(e106, [e, =, 6], 167, 1, []),
event(e107, [f, =, 6], 168, 2, []),
event(e108, [f, =, 6], 170, 3, []),
event(e109, [d, =, 6], 173, 1, []),
event(e110, [e, =, 6], 174, 1, []),
event(e111, [f, =, 6], 175, 1, []),
event(e112, [f, =, 6], 176, 2, []),
event(e113, [d, =, 6], 178, 3, []),
```

```
event(e114, [a, =, 5], 181, 1, []),
event(e115, [b, =, 5], 182, 1, []),
event(e116, [d, =, 6], 183, 1, []),
event(e117, [c, =, 6], 184, 1, []),
event(e118, [e, =, 6], 185, 1, []),
event(e119, [a, =, 6], 186, 3, []),
event(e120, [f, =, 7], 189, 1, []),
event(e121, [a, =, 7], 190, 3, []),
event(e122, [c, =, 6], 193, 1, []),
event(e123, [e, =, 6], 194, 1, []),
event(e124, [a, =, 6], 195, 1, []),
event(e125, [g, =, 6], 196, 1, []),
event(e126, [f, #, 6], 197, 1, []),
event(e127, [e, =, 6], 198, 1, []),
event(e128, [f, #, 6], 199, 1, []),
event(e129, [g, =, 6], 200, 2, []),
event(e130, [g, =, 6], 202, 3, []),
event(e131, [e, =, 6], 205, 1, []),
event(e132, [f, #, 6], 206, 1, []),
event(e133, [g, =, 6], 207, 1, []),
event(e134, [g, =, 6], 208, 2, []),
event(e135, [e, =, 6], 210, 3, []),
event(e136, [b, =, 5], 213, 1, []),
event(e137, [c, #, 6], 214, 1, []),
event(e138, [e, =, 6], 215, 1, []),
event(e139, [d, =, 6], 216, 1, []),
event(e140, [f, #, 6], 217, 1, []),
event(e141, [b, =, 6], 218, 3, []),
event(e142, [g, =, 7], 221, 1, []),
event(e143, [b, =, 7], 222, 3, []),
event(e144, [f, #, 7], 225, 1, []),
event(e145, [b, =, 7], 226, 1, []),
event(e146, [a, =, 7], 227, 1, []),
event(e147, [g, #, 7], 228, 1, []),
event(e148, [f, #, 7], 229, 1, []),
event(e149, [d, #, 7], 230, 1, []),
event(e150, [b, =, 6], 231, 1, []),
event(e151, [g, #, 6], 232, 0.5, []),
event(e152, [b, =, 6], 232.5, 0.5, []),
event(e153, [e, =, 7], 233, 0.5, []),
event(e154, [c, #, 7], 233.5, 0.5, []),
event(e155, [d, #, 7], 234, 2, []),
event(e156, [g, #, 6], 236, 0.5, []),
event(e157, [b, =, 6], 236.5, 0.5, []),
```

```
event(e158, [e, =, 7], 237, 0.5, []),
event(e159, [c, #, 7], 237.5, 0.5, []),
event(e160, [d, #, 7], 238, 2, []),
event(e161, [g, #, 6], 240, 0.5, []),
event(e162, [b, =, 6], 240.5, 0.5, []),
event(e163, [e, =, 7], 241, 0.5, []),
event(e164, [c, #, 7], 241.5, 0.5, []),
event(e165, [d, #, 7], 242, 2, []),
event(e166, [g, #, 6], 244, 0.5, []),
event(e167, [b, =, 6], 244.5, 0.5, []),
event(e168, [e, =, 7], 245, 0.5, []),
event(e169, [c, #, 7], 245.5, 0.5, []),
event(e170, [d, #, 7], 246, 2, []),
event(e171, [g, #, 6], 248, 0.5, []),
event(e172, [b, =, 6], 248.5, 0.5, []),
event(e173, [e, =, 7], 249, 0.5, []),
event(e174, [c, #, 7], 249.5, 0.5, []),
event(e175, [d, #, 7], 250, 2, []),
event(e176, [g, #, 6], 252, 0.5, []),
event(e177, [b, =, 6], 252.5, 0.5, []),
event(e178, [e, =, 7], 253, 0.5, []),
event(e179, [c, #, 7], 253.5, 0.5, []),
event(e180, [d, #, 7], 254, 2, []),
event(e181, [g, #, 6], 256, 0.5, []),
event(e182, [b, =, 6], 256.5, 0.5, []),
event(e183, [e, =, 7], 257, 0.5, []),
event(e184, [c, #, 7], 257.5, 0.5, []),
event(e185, [d, #, 7], 258, 2, []),
event(e186, [g, #, 6], 260, 0.5, []),
event(e187, [b, =, 6], 260.5, 0.5, []),
event(e188, [e, =, 7], 261, 0.5, []),
event(e189, [c, #, 7], 261.5, 0.5, []),
event(e190, [e, =, 7], 262, 2, []),
event(e191, [c, #, 7], 264, 0.5, []),
event(e192, [e, =, 7], 264.5, 0.5, []),
event(e193, [a, =, 7], 265, 0.5, []),
event(e194, [f, #, 7], 265.5, 0.5, []),
event(e195, [g, #, 7], 266, 2, []),
event(e196, [c, #, 7], 268, 0.5, []),
event(e197, [e, =, 7], 268.5, 0.5, []),
event(e198, [a, =, 7], 269, 0.5, []),
event(e199, [f, #, 7], 269.5, 0.5, []),
event(e200, [g, #, 7], 270, 2, []),
event(e201, [c, #, 7], 272, 0.5, []),
```

```
event(e202, [e, =, 7], 272.5, 0.5, []),
event(e203, [a, =, 7], 273, 0.5, []),
event(e204, [f, #, 7], 273.5, 0.5, []),
event(e205, [g, #, 7], 274, 2, []),
event(e206, [c, #, 7], 276, 0.5, []),
event(e207, [e, =, 7], 276.5, 0.5, []),
event(e208, [a, =, 7], 277, 0.5, []),
event(e209, [e, =, 7], 277.5, 0.5, []),
event(e210, [g, =, 7], 278, 2, []),
event(e211, [e, =, 6], 280, 0.5, []),
event(e212, [g, #, 6], 280.5, 0.5, []),
event(e213, [b, =, 6], 281, 0.5, []),
event(e214, [g, #, 6], 281.5, 0.5, []),
event(e215, [c, #, 7], 282, 2, []),
event(e216, [f, #, 6], 284, 0.5, []),
event(e217, [a, =, 6], 284.5, 0.5, []),
event(e218, [d, =, 7], 285, 0.5, []),
event(e219, [a, =, 6], 285.5, 0.5, []),
event(e220, [d, #, 7], 286, 2, []),
event(e221, [e, =, 7], 288, 1, []),
event(e222, [b, =, 6], 290, 3, []),
event(e223, [g, #, 6], 293, 1, []),
event(e224, [a, =, 6], 294, 1, []),
event(e225, [b, =, 6], 295, 1, []),
event(e226, [g, #, 6], 296, 1, []),
event(e227, [e, =, 6], 297, 1, []),
event(e228, [f, #, 6], 298, 1, []),
event(e229, [g, #, 6], 299, 1, []),
event(e230, [e, =, 6], 300, 2, []),
event(e231, [b, =, 5], 302, 1, []),
event(e232, [g, #, 6], 304, 0.5, []),
event(e233, [b, =, 6], 304.5, 0.5, []),
event(e234, [e, =, 7], 305, 0.5, []),
event(e235, [c, #, 7], 305.5, 0.5, []),
event(e236, [d, #, 7], 306, 2, []),
event(e237, [g, #, 6], 308, 0.5, []),
event(e238, [b, =, 6], 308.5, 0.5, []),
event(e239, [e, =, 7], 309, 0.5, []),
event(e240, [c, #, 7], 309.5, 0.5, []),
event(e241, [d, #, 7], 310, 2, []),
event(e242, [g, #, 6], 312, 0.5, []),
event(e243, [b, =, 6], 312.5, 0.5, []),
event(e244, [e, =, 7], 313, 0.5, []),
event(e245, [c, #, 7], 313.5, 0.5, []),
```

```
event(e246, [d, #, 7], 314, 2, []),
event(e247, [g, #, 6], 316, 0.5, []),
event(e248, [b, =, 6], 316.5, 0.5, []),
event(e249, [e, =, 7], 317, 0.5, []),
event(e250, [c, #, 7], 317.5, 0.5, []),
event(e251, [d, #, 7], 318, 2, []),
event(e252, [g, #, 6], 320, 0.5, []),
event(e253, [b, =, 6], 320.5, 0.5, []),
event(e254, [e, =, 7], 321, 0.5, []),
event(e255, [c, #, 7], 321.5, 0.5, []),
event(e256, [d, #, 7], 322, 2, []),
event(e257, [g, #, 6], 324, 0.5, []),
event(e258, [b, =, 6], 324.5, 0.5, []),
event(e259, [e, =, 7], 325, 0.5, []),
event(e260, [c, #, 7], 325.5, 0.5, []),
event(e261, [d, #, 7], 326, 2, []),
event(e262, [e, =, 7], 328, 2, []),

constituent(c001, stream, faure, [e001, e002, e003, e004,
e005, e006, e007, e008, e009, e010, e011, e012, e013, e014,
e015, e016, e017, e018, e019, e020, e021, e022, e023, e024,
e025, e026, e027, e028, e029, e030, e031, e032, e033, e034,
e035, e036, e037, e038, e039, e040, e041, e042, e043, e044,
e045, e046, e047, e048, e049, e050, e051, e052, e053, e054,
e055, e056, e057, e058, e059, e060, e061, e062, e063, e064,
e065, e066, e067, e068, e069, e070, e071, e072, e073, e074,
e075, e076, e077, e078, e079, e080, e081, e082, e083, e084,
e085, e086, e087, e088, e089, e090, e091, e092, e093, e094,
e095, e096, e097, e098, e099, e100, e101, e102, e103, e104,
e105, e106, e107, e108, e109, e110, e111, e112, e113, e114,
e115, e116, e117, e118, e119, e120, e121, e122, e123, e124,
e125, e126, e127, e128, e129, e130, e131, e132, e133, e134,
e135, e136, e137, e138, e139, e140, e141, e142, e143, e144,
e145, e146, e147, e148, e149, e150, e151, e152, e153, e154,
e155, e156, e157, e158, e159, e160, e161, e162, e163, e164,
e165, e166, e167, e168, e169, e170, e171, e172, e173, e174,
e175, e176, e177, e178, e179, e180, e181, e182, e183, e184,
e185, e186, e187, e188, e189, e190, e191, e192, e193, e194,
e195, e196, e197, e198, e199, e200, e201, e202, e203, e204,
e205, e206, e207, e208, e209, e210, e211, e212, e213, e214,
e215, e216, e217, e218, e219, e220, e221, e222, e223, e224,
e225, e226, e227, e228, e229, e230, e231, e232, e233, e234,
e235, e236, e237, e238, e239, e240, e241, e242, e243, e244,
e245, e246, e247, e248, e249, e250, e251, e252, e253, e254,
```

```
e255, e256, e257, e258, e259, e260, e261, e262]),

constituent(c002, dynamics, piano, [e001, e002, e003, e004,
  e005, e006, e007, e008, e009, e010, e011, e012, e013, e014,
  e015, e016]),
constituent(c003, dynamics, crescendo, [e017, e018, e019,
  e020]),
constituent(c004, dynamics, decrescendo, [e021, e022]),
constituent(c005, dynamics, piano, [e023, e024, e025, e026,
  e027, e028, e029, e030, e031, e032, e033]),
constituent(c006, dynamics, crescendo, [e034, e035, e036,
  e037, e038, e039, e040, e041, e042]),
constituent(c007, dynamics, decrescendo, [e043, e044]),
constituent(c008, dynamics, piano, [e045, e046, e047, e048,
  e049, e050, e051, e052, e053, e054, e055, e056, e057, e058,
  e059, e060, e061, e062, e063, e064, e065, e066]),
constituent(c009, dynamics, crescendo, [e067, e068, e069,
  e070, e071, e072, e073, e074, e075, e076, e077]),
constituent(c010, dynamics, forte, [e078, e079, e080, e081]),
constituent(c011, dynamics, piano, [e084]),
constituent(c012, dynamics, sempredolce, [e085, e086, e087,
  e088, e089, e090, e091, e092, e093, e094, e095, e096, e097,
  e098, e099, e100, e101, e102, e103, e104, e105, e106, e107,
  e108, e109, e110, e111, e112, e113, e114, e115, e116, e117,
  e118, e119, e120, e121, e122, e123, e124, e125, e126, e127,
  e128]),
constituent(c013, dynamics, crescendo, [e129, e130, e131,
  e132, e133, e134, e135, e136, e137, e138, e139, e140, e141,
  e142, e143]),
constituent(c014, dynamics, forte, [e144, e145, e146]),
constituent(c015, dynamics, decrescendo, [e147, e148, e149,
  e150]),
constituent(c016, dynamics, pianissimo, [e151, e152, e153,
  e154, e155, e156, e157, e158, e159, e160, e161, e162, e163,
  e164, e165, e166, e167, e168, e169, e170, e171, e172, e173,
  e174, e175, e176, e177, e178, e179, e180, e181, e182, e183,
  e184, e185, e186, e187, e188, e189, e190, e191, e192, e193,
  e194, e195, e196, e197, e198, e199, e200, e201, e202, e203,
  e204, e205, e206, e207, e208, e209, e210, e211, e212, e213,
  e214, e215, e216, e217, e218, e219, e220, e221]),
constituent(c017, dynamics, piano, [e222, e223, e224, e225,
  e226, e227, e228, e229, e230, e231]),
constituent(c018, dynamics, pianissimo, [e232, e233, e234,
  e235, e236, e237, e238, e239, e240, e241, e242, e243, e244,
```

```
  e245, e246, e247, e248, e249, e250, e251, e252, e253, e254,
  e255, e256, e257, e258, e259, e260, e261, e262]),

constituent(c019, articulation, slur, [e003, e004, e005, e006,
  e007, e008, e009]),
constituent(c020, articulation, slur, [e010, e011]),
constituent(c021, articulation, slur, [e014, e015, e016, e017,
  e018, e019, e020]),
constituent(c022, articulation, slur, [e021, e022]),
constituent(c023, articulation, slur, [e025, e026, e027, e028,
  e029, e030, e031]),
constituent(c024, articulation, slur, [e032, e033]),
constituent(c025, articulation, slur, [e036, e037, e038, e039,
  e040, e041, e042]),
constituent(c026, articulation, slur, [e043, e044]),
constituent(c025, articulation, slur, [e047, e048, e049, e050,
  e051, e052, e053]),
constituent(c026, articulation, slur, [e054, e055]),
constituent(c027, articulation, slur, [e058, e059, e060, e061,
  e062, e063, e064]),
constituent(c028, articulation, slur, [e065, e066]),
constituent(c029, articulation, slur, [e069, e070, e071]),
constituent(c030, articulation, slur, [e074, e075, e076]),
constituent(c031, articulation, slur, [e079, e080, e081, e082,
  e083, e084]),
constituent(c032, articulation, slur, [e087, e088, e089, e090,
  e091]),
constituent(c033, articulation, slur, [e092, e093, e094, e095,
  e096]),
constituent(c034, articulation, slur, [e098, e099]),
constituent(c035, articulation, slur, [e100, e101, e102, e103,
  e104, e105, 106]),
constituent(c036, articulation, slur, [e109, e110, e111, e112,
  e113]),
constituent(c037, articulation, slur, [e114, e115, e116, e117,
  e118]),
constituent(c038, articulation, slur, [e120, e121]),
constituent(c039, articulation, slur, [e122, e123, e124, e125,
  e126, e127, e128]),
constituent(c040, articulation, slur, [e131, e132, e133]),
constituent(c041, articulation, slur, [e136, e137, e138, e139,
  e140]),
constituent(c042, articulation, slur, [e144, e145, e146, e147,
  e148, e149, e150]),
```

```
        constituent(c043, articulation, slur, [e151, e152, e153, e154,
          e155]),
        constituent(c044, articulation, slur, [e156, e157, e158, e159,
          e160]),
        constituent(c045, articulation, slur, [e161, e162, e163, e164,
          e165]),
        constituent(c046, articulation, slur, [e166, e167, e168, e169,
          e170]),
        constituent(c047, articulation, slur, [e171, e172, e173, e174,
          e175]),
        constituent(c048, articulation, slur, [e176, e177, e178, e179,
          e180]),
        constituent(c049, articulation, slur, [e181, e182, e183, e184,
          e185]),
        constituent(c050, articulation, slur, [e186, e187, e188, e189,
          e190]),
        constituent(c051, articulation, slur, [e191, e192, e193, e194,
          e195]),
        constituent(c052, articulation, slur, [e196, e197, e198, e199,
          e200]),
        constituent(c053, articulation, slur, [e201, e202, e203, e204,
          e205]),
        constituent(c054, articulation, slur, [e206, e207, e208, e209,
          e210]),
        constituent(c053, articulation, slur, [e211, e212, e213, e214,
          e215, e216, e217, e218, e219, e220]),
        constituent(c054, articulation, slur, [e223, e224, e225, e226,
          e227, e228, e229, e230, e231]),
        constituent(c055, articulation, slur, [e232, e233, e234, e235,
          e236]),
        constituent(c056, articulation, slur, [e237, e238, e239, e240,
          e241]),
        constituent(c055, articulation, slur, [e242, e243, e244, e245,
          e246]),
        constituent(c056, articulation, slur, [e247, e248, e249, e250,
          e251]),
        constituent(c055, articulation, slur, [e252, e253, e254, e255,
          e256]),
        constituent(c056, articulation, slur, [e257, e258, e259, e260,
          e261])
]).
```

A.2 Beethoven's *Auf dem Hügel sitz ich spähend*

```
piece(beethoven, [
        event(e001, [e, =, 6], 002, 2, []),
        event(e002, [e, =, 6], 004, 2, []),
        event(e003, [e, =, 6], 006, 3, []),
        event(e004, [f, #, 6], 009, 1, []),
        event(e005, [g, #, 6], 010, 1, []),
        event(e006, [a, =, 6], 011, 1, []),
        event(e007, [a, =, 6], 012, 2, []),
        event(e008, [c, #, 6], 014, 1, []),
        event(e009, [d, =, 6], 016, 1, []),
        event(e010, [c, #, 6], 017, 1, []),
        event(e011, [b, =, 5], 018, 1, []),
        event(e012, [d, =, 6], 019, 1, []),
        event(e013, [f, #, 6], 020, 2, []),
        event(e014, [f, #, 6], 022, 2, []),
        event(e015, [b, =, 5], 024, 2, []),
        event(e016, [e, =, 6], 026, 3, []),
        event(e017, [f, #, 6], 029, 1, []),
        event(e018, [e, =, 6], 030, 2, []),
        event(e019, [d, #, 6], 032, 2, []),
        event(e020, [d, =, 6], 034, 1, []),
        event(e021, [e, =, 6], 035, 1, []),
        event(e022, [d, =, 6], 036, 2, []),
        event(e023, [c, #, 6], 038, 2, []),
        event(e024, [a, =, 6], 040, 1, []),
        event(e025, [g, #, 6], 041, 1, []),
        event(e026, [g, #, 6], 042, 2, []),
        event(e027, [f, #, 6], 044, 1, []),
        event(e028, [e, =, 6], 045, 1, []),
        event(e029, [d, =, 6], 046, 1, []),
        event(e030, [b, =, 5], 047, 1, []),
        event(e031, [a, =, 6], 048, 2, []),
        event(e032, [e, =, 6], 062, 2, []),
        event(e033, [e, =, 6], 064, 2, []),
        event(e034, [e, =, 6], 066, 3, []),
        event(e035, [f, #, 6], 069, 1, []),
        event(e036, [g, #, 6], 070, 1, []),
        event(e037, [a, =, 6], 071, 1, []),
        event(e038, [a, =, 6], 072, 2, []),
        event(e039, [c, #, 6], 074, 1, []),
        event(e040, [d, =, 6], 076, 1, []),
        event(e041, [c, #, 6], 077, 1, []),
```

```
event(e042, [b, =, 5], 078, 1, []),
event(e043, [d, =, 6], 079, 1, []),
event(e044, [f, #, 6], 080, 2, []),
event(e045, [f, #, 6], 082, 2, []),
event(e046, [b, =, 5], 084, 2, []),
event(e047, [e, =, 6], 086, 3, []),
event(e048, [f, #, 6], 089, 1, []),
event(e049, [e, =, 6], 090, 2, []),
event(e050, [e, =, 6], 092, 2, []),
event(e051, [d, =, 6], 094, 1, []),
event(e052, [e, =, 6], 095, 1, []),
event(e053, [d, =, 6], 096, 2, []),
event(e054, [c, #, 6], 098, 2, []),
event(e055, [a, =, 6], 100, 1, []),
event(e056, [g, #, 6], 101, 1, []),
event(e057, [g, #, 6], 102, 2, []),
event(e058, [f, #, 6], 104, 1, []),
event(e059, [f, =, 6], 105, 1, []),
event(e060, [d, =, 6], 106, 1, []),
event(e061, [b, =, 5], 107, 1, []),
event(e062, [a, =, 6], 108, 2, []),
event(e063, [e, =, 6], 122, 2, []),
event(e064, [e, =, 6], 124, 2, []),
event(e065, [e, =, 6], 126, 3, []),
event(e066, [f, #, 6], 129, 1, []),
event(e067, [g, #, 6], 130, 1, []),
event(e068, [a, =, 6], 131, 1, []),
event(e069, [a, =, 6], 132, 2, []),
event(e070, [c, #, 6], 134, 1, []),
event(e071, [d, =, 6], 136, 1, []),
event(e072, [c, #, 6], 137, 1, []),
event(e073, [b, =, 5], 138, 1, []),
event(e074, [d, =, 6], 139, 1, []),
event(e075, [f, #, 6], 140, 1, []),
event(e076, [f, =, 5], 141, 2, []),
event(e077, [f, #, 6], 143, 1, []),
event(e078, [b, =, 5], 144, 2, []),
event(e079, [e, =, 6], 146, 3, []),
event(e080, [f, #, 6], 149, 1, []),
event(e081, [e, =, 6], 150, 2, []),
event(e082, [e, =, 6], 152, 2, []),
event(e083, [d, =, 6], 154, 1, []),
event(e084, [e, =, 6], 155, 1, []),
event(e085, [d, =, 6], 156, 2, []),
```

```
event(e086, [c, #, 6], 158, 1, []),
event(e087, [a, =, 6], 160, 1, []),
event(e088, [g, #, 6], 161, 1, []),
event(e089, [g, #, 6], 162, 2, []),
event(e090, [f, #, 6], 164, 1, []),
event(e091, [e, =, 6], 165, 1, []),
event(e092, [d, =, 6], 166, 1, []),
event(e093, [b, =, 5], 167, 1, []),
event(e094, [a, =, 6], 168, 2, []),
event(e095, [e, =, 6], 182, 2, []),
event(e096, [e, =, 6], 184, 2, []),
event(e097, [e, =, 6], 186, 3, []),
event(e098, [f, #, 6], 189, 1, []),
event(e099, [g, #, 6], 190, 1, []),
event(e100, [a, =, 6], 191, 1, []),
event(e101, [a, =, 6], 192, 2, []),
event(e102, [c, #, 6], 194, 1, []),
event(e103, [d, =, 6], 196, 1, []),
event(e104, [c, #, 6], 197, 1, []),
event(e105, [b, =, 5], 198, 1, []),
event(e106, [d, =, 6], 199, 1, []),
event(e107, [f, #, 6], 200, 2, []),
event(e108, [f, =, 5], 202, 1, []),
event(e109, [f, #, 6], 203, 1, []),
event(e110, [b, =, 5], 204, 2, []),
event(e111, [e, =, 6], 206, 3, []),
event(e112, [f, #, 6], 209, 1, []),
event(e113, [e, =, 6], 210, 2, []),
event(e114, [e, =, 6], 212, 2, []),
event(e115, [d, =, 6], 214, 0.7, []),
event(e116, [f, #, 6], 214.7, 0.6, []),
event(e117, [e, =, 6], 215.3, 0.7, []),
event(e118, [d, =, 6], 216, 2, []),
event(e119, [c, #, 6], 218, 2, []),
event(e120, [a, =, 6], 220, 1, []),
event(e121, [g, #, 6], 221, 1, []),
event(e122, [g, #, 6], 222, 2, []),
event(e123, [f, #, 6], 224, 1, []),
event(e124, [e, =, 6], 225, 1, []),
event(e125, [d, =, 6], 226, 1, []),
event(e126, [b, =, 5], 227, 1, []),
event(e127, [a, =, 6], 228, 2, []),
event(e128, [e, =, 6], 242, 2, []),
event(e129, [e, =, 6], 244, 2, []),
```

```
event(e130, [e, =, 6], 246, 3, []),
event(e131, [f, #, 6], 249, 1, []),
event(e132, [g, #, 6], 250, 1, []),
event(e133, [a, =, 6], 251, 1, []),
event(e134, [a, =, 6], 252, 2, []),
event(e135, [c, #, 6], 254, 1, []),
event(e136, [d, =, 6], 256, 1, []),
event(e137, [c, #, 6], 257, 1, []),
event(e138, [b, =, 5], 258, 1, []),
event(e139, [d, =, 6], 259, 1, []),
event(e140, [f, #, 6], 260, 2, []),
event(e141, [f, #, 6], 262, 2, []),
event(e142, [b, =, 5], 264, 2, []),
event(e143, [e, =, 6], 266, 3, []),
event(e144, [f, #, 6], 269, 1, []),
event(e145, [e, =, 6], 270, 2, []),
event(e146, [e, =, 6], 272, 1, []),
event(e147, [d, =, 6], 273, 1, []),
event(e148, [d, =, 6], 274, 1, []),
event(e149, [e, =, 6], 275, 1, []),
event(e150, [d, =, 6], 276, 2, []),
event(e151, [c, #, 6], 278, 2, []),
event(e152, [a, =, 6], 280, 1, []),
event(e153, [g, #, 6], 281, 1, []),
event(e154, [g, #, 6], 282, 2, []),
event(e155, [f, #, 6], 284, 1, []),
event(e156, [e, =, 6], 285, 1, []),
event(e157, [d, =, 6], 286, 1, []),
event(e158, [b, =, 5], 287, 1, []),
event(e159, [a, =, 6], 288, 2, []),

constituent(c001, stream, beethoven, [e001, e002, e003, e004,
  e005, e006, e007, e008, e009, e010, e011, e012, e013, e014,
  e015, e016, e017, e018, e019, e020, e021, e022, e023, e024,
  e025, e026, e027, e028, e029, e030, e031, e032, e033, e034,
  e035, e036, e037, e038, e039, e040, e041, e042, e043, e044,
  e045, e046, e047, e048, e049, e050, e051, e052, e053, e054,
  e055, e056, e057, e058, e059, e060, e061, e062, e063, e064,
  e065, e066, e067, e068, e069, e070, e071, e072, e073, e074,
  e075, e076, e077, e078, e079, e080, e081, e082, e083, e084,
  e085, e086, e087, e088, e089, e090, e091, e092, e093, e094,
  e095, e096, e097, e098, e099, e100, e101, e102, e103, e104,
  e105, e106, e107, e108, e109, e110, e111, e112, e113, e114,
  e115, e116, e117, e118, e119, e120, e121, e122, e123, e124,
```

```
  e125, e126, e127, e128, e129, e130, e131, e132, e133, e134,
  e135, e136, e137, e138, e139, e140, e141, e142, e143, e144,
  e145, e146, e147, e148, e149, e150, e151, e152, e153, e154,
  e155, e156, e157, e158, e159]),

constituent(c002, dynamics, piano, [e001, e002, e003, e004,
  e005, e006, e007, e008, e009, e010, e011, e012, e013, e014,
  e015, e016, e017, e018, e019, e020, e021, e022, e023]),
constituent(c003, dynamics, crescendo, [e024, e025]),
constituent(c004, dynamics, decrescendo, [e026, e027]),
constituent(c005, dynamics, diminuendo, [e032, e033, e034,
  e035, e036, e037, e038, e039, e040, e041, e042, e043, e044,
  e045, e046, e047, e048, e049]),
constituent(c006, dynamics, crescendo, [e050, e051, e052,
  e053, e054, e055, e056]),
constituent(c007, dynamics, decrescendo, [e057, e058]),
constituent(c008, dynamics, diminuendo, [e063, e064, e065,
  e066, e067, e068, e069, e070, e071, e072]),
constituent(c009, dynamics, crescendo, [e073, e074, e075,
  e076, e077]),
constituent(c010, dynamics, piano, [e078, e079, e080, e081,
  e082, e083, e084, e085, e086, e087, e088, e089, e090, e091,
  e092, e093, e094]),
constituent(c011, dynamics, swell, [e098, e099, e100]),
constituent(c012, dynamics, swell, [e101, e102, e103, e104]),
constituent(c013, dynamics, piano, [e105, e106, e107, e108,
  e109, e110]),
constituent(c014, dynamics, dolce, [e111, e112, e113, e114,
  e115, e116, e117, e118, e119]),
constituent(c015, dynamics, crescendo, [e120, e121]),
constituent(c016, dynamics, piano, [e121, e122, e123, e124,
  e125, e126, e127]),
constituent(c017, dynamics, piano, [e128, e129, e130, e131,
  e132, e133, e134, e135, e136, e137, e138, e139, e140]),
constituent(c018, dynamics, crescendo, [e141, e142, e143,
  e144, e145, e146, e147, e148, e149, e150, e151, e152, e153,
  e154, e155, e156, e157, e158]),
constituent(c019, dynamics, forte, [e159]),

constituent(c020, articulation, slur, [e026, e027]),
constituent(c021, articulation, slur, [e057, e058]),
constituent(c022, articulation, slur, [e075, e076]),
constituent(c023, articulation, slur, [e089, e090]),
constituent(c024, articulation, slur, [e107, e108]),
```

```
        constituent(c025, articulation, triplet, [e115, e116, e117]),
        constituent(c026, articulation, slur, [e122, e123]),
        constituent(c027, articulation, slur, [e145, e146]),
        constituent(c028, articulation, slur, [e154, e155])
]).
```

A.3 Schubert's *Gute Nacht*

```
piece(schubert, [
        event(e001, [f, =, 6], 013.5, 0.5, []),
        event(e002, [e, =, 6], 014, 0.5, []),
        event(e003, [d, =, 6], 014.5, 0.5, []),
        event(e004, [a, =, 5], 015, 0.5, []),
        event(e005, [f, =, 5], 015.5, 0.5, []),
        event(e006, [e, =, 5], 016, 0.75, []),
        event(e007, [f, =, 5], 016.75, 0.25, []),
        event(e008, [e, =, 5], 017, 0.5, []),
        event(e009, [d, =, 6], 017.5, 0.5, []),
        event(e010, [a, =, 5], 018, 0.75, []),
        event(e011, [f, =, 5], 018.75, 0.25, []),
        event(e012, [a, =, 5], 019, 0.25, []),
        event(e013, [f, =, 5], 019.25, 0.25, []),
        event(e014, [e, =, 5], 019.5, 0.25, []),
        event(e015, [f, =, 5], 019.75, 0.25, []),
        event(e016, [d, =, 5], 020, 1.0, []),
        event(e017, [f, =, 6], 021.5, 0.5, []),
        event(e018, [e, =, 6], 022, 0.5, []),
        event(e019, [d, =, 6], 022.5, 0.5, []),
        event(e020, [a, =, 5], 023, 0.5, []),
        event(e021, [f, =, 5], 023.5, 0.5, []),
        event(e022, [e, =, 5], 024, 0.75, []),
        event(e023, [f, =, 5], 024.75, 0.25, []),
        event(e024, [e, =, 5], 025, 0.5, []),
        event(e025, [d, =, 6], 025.5, 0.5, []),
        event(e026, [a, =, 5], 026, 0.75, []),
        event(e027, [f, =, 5], 026.75, 0.25, []),
        event(e028, [a, =, 5], 027, 0.25, []),
        event(e029, [f, =, 5], 027.25, 0.25, []),
        event(e030, [e, =, 5], 027.5, 0.25, []),
        event(e031, [f, =, 5], 027.75, 0.25, []),
        event(e032, [d, =, 5], 028, 1.0, []),
        event(e033, [c, =, 5], 029.5, 0.5, []),
        event(e034, [f, =, 5], 030, 0.5, []),
        event(e035, [f, =, 5], 030.5, 0.5, []),
        event(e036, [f, =, 5], 031, 0.5, []),
        event(e037, [f, =, 5], 031.5, 0.5, []),
        event(e038, [g, =, 5], 032, 0.75, []),
        event(e039, [a, =, 5], 032.75, 0.25, []),
        event(e040, [g, =, 5], 033, 0.5, []),
        event(e041, [a, =, 5], 033.5, 0.5, []),
```

```
event(e042, [c, =, 6], 034, 0.25, []),
event(e043, [a, #, 5], 034.25, 0.25, []),
event(e044, [a, =, 5], 034.5, 0.5, []),
event(e045, [g, =, 5], 035, 0.5, []),
event(e046, [a, =, 5], 035.5, 0.25, []),
event(e047, [a, #, 5], 035.75, 0.25, []),
event(e048, [a, =, 5], 036, 1.0, []),
event(e049, [f, =, 5], 037.5, 0.5, []),
event(e050, [a, #, 5], 038, 0.5, []),
event(e051, [a, #, 5], 038.5, 0.5, []),
event(e052, [a, #, 5], 039, 0.5, []),
event(e053, [a, #, 5], 039.5, 0.5, []),
event(e054, [c, =, 6], 040, 0.75, []),
event(e055, [d, =, 6], 040.75, 0.25, []),
event(e056, [c, =, 6], 041, 0.5, []),
event(e057, [d, =, 6], 041.5, 0.5, []),
event(e058, [f, =, 6], 042, 0.25, []),
event(e059, [d, #, 6], 042.25, 0.25, []),
event(e060, [d, =, 6], 042.5, 0.5, []),
event(e061, [c, =, 6], 043, 0.75, []),
event(e062, [d, =, 6], 043.75, 0.25, []),
event(e063, [a, #, 5], 044, 1.0, []),
event(e064, [d, =, 6], 049.5, 0.5, []),
event(e065, [a, #, 5], 050, 0.75, []),
event(e066, [g, =, 5], 050.75, 0.25, []),
event(e067, [e, =, 5], 051, 0.5, []),
event(e068, [d, =, 6], 051.5, 0.5, []),
event(e069, [a, =, 5], 052, 0.75, []),
event(e070, [f, =, 5], 052.75, 0.25, []),
event(e071, [d, =, 5], 053, 0.5, []),
event(e072, [f, =, 5], 053.5, 0.25, []),
event(e073, [g, =, 5], 053.75, 0.25, []),
event(e074, [a, =, 5], 054, 0.5, []),
event(e075, [a, =, 5], 054.5, 0.5, []),
event(e076, [a, =, 5], 055, 0.5, []),
event(e077, [b, =, 5], 055.5, 0.25, []),
event(e078, [c, #, 6], 055.75, 0.25, []),
event(e079, [d, =, 6], 056, 1.0, []),
event(e080, [d, =, 6], 057.5, 0.5, []),
event(e081, [a, #, 5], 058, 0.75, []),
event(e082, [g, =, 5], 058.75, 0.25, []),
event(e083, [e, =, 5], 059, 0.5, []),
event(e084, [d, =, 6], 059.5, 0.5, []),
event(e085, [a, =, 5], 060, 0.75, []),
```

```
event(e086, [f, =, 5], 060.75, 0.25, []),
event(e087, [d, =, 5], 061, 0.5, []),
event(e088, [f, =, 5], 061.5, 0.25, []),
event(e089, [g, =, 5], 061.75, 0.25, []),
event(e090, [a, =, 5], 062, 0.5, []),
event(e091, [a, =, 5], 062.5, 0.5, []),
event(e092, [a, =, 5], 063, 0.5, []),
event(e093, [a, =, 5], 063.5, 0.5, []),
event(e094, [d, =, 5], 064, 1.0, []),
event(e095, [f, =, 6], 077.5, 0.5, []),
event(e096, [e, =, 6], 078, 0.5, []),
event(e097, [d, =, 6], 078.5, 0.5, []),
event(e098, [a, =, 5], 079, 0.5, []),
event(e099, [f, =, 5], 079.5, 0.5, []),
event(e100, [e, =, 5], 080, 0.75, []),
event(e101, [g, =, 5], 080.75, 0.25, []),
event(e102, [a, #, 5], 081, 0.5, []),
event(e103, [d, =, 6], 081.5, 0.5, []),
event(e104, [a, =, 5], 082, 0.75, []),
event(e105, [f, =, 5], 082.75, 0.25, []),
event(e106, [g, =, 5], 083, 0.25, []),
event(e107, [a, =, 5], 083.25, 0.25, []),
event(e108, [b, =, 5], 083.5, 0.25, []),
event(e109, [c, #, 6], 083.75, 0.25, []),
event(e110, [d, =, 6], 084, 1.0, []),
event(e111, [f, =, 6], 085.5, 0.5, []),
event(e112, [e, =, 6], 086, 0.5, []),
event(e113, [d, =, 6], 086.5, 0.5, []),
event(e114, [a, =, 5], 087, 0.5, []),
event(e115, [f, =, 5], 087.5, 0.5, []),
event(e116, [e, =, 5], 088, 0.75, []),
event(e117, [g, =, 5], 088.75, 0.25, []),
event(e118, [a, #, 5], 089, 0.5, []),
event(e119, [d, =, 6], 089.5, 0.5, []),
event(e120, [a, =, 5], 090, 0.75, []),
event(e121, [f, =, 5], 090.75, 0.25, []),
event(e122, [g, =, 5], 091, 0.25, []),
event(e123, [a, =, 5], 091.25, 0.25, []),
event(e124, [b, =, 5], 091.5, 0.25, []),
event(e125, [c, #, 6], 091.75, 0.25, []),
event(e126, [d, =, 6], 092, 1.0, []),
event(e127, [e, =, 5], 093.5, 0.5, []),
event(e128, [f, =, 5], 094, 0.5, []),
event(e129, [f, =, 5], 094.5, 0.5, []),
```

```
event(e130, [f, =, 5], 095, 0.5, []),
event(e131, [f, =, 5], 095.5, 0.5, []),
event(e132, [g, =, 5], 096, 0.75, []),
event(e133, [a, =, 5], 096.75, 0.25, []),
event(e134, [g, =, 5], 097, 0.5, []),
event(e135, [a, =, 5], 097.5, 0.5, []),
event(e136, [c, =, 6], 098, 0.25, []),
event(e137, [a, #, 5], 098.25, 0.25, []),
event(e138, [a, =, 5], 098.5, 0.5, []),
event(e139, [g, =, 5], 099, 0.5, []),
event(e140, [a, =, 5], 099.5, 0.25, []),
event(e141, [a, #, 5], 099.75, 0.25, []),
event(e142, [a, =, 5], 100, 1.0, []),
event(e143, [f, =, 5], 101.5, 0.5, []),
event(e144, [a, #, 5], 102, 0.5, []),
event(e145, [a, #, 5], 102.5, 0.5, []),
event(e146, [a, #, 5], 103, 0.5, []),
event(e147, [a, #, 5], 103.5, 0.5, []),
event(e148, [c, =, 6], 104, 0.75, []),
event(e149, [d, =, 6], 104.75, 0.25, []),
event(e150, [c, =, 6], 105, 0.5, []),
event(e151, [d, =, 6], 105.5, 0.5, []),
event(e152, [f, =, 6], 106, 0.25, []),
event(e153, [d, #, 6], 106.25, 0.25, []),
event(e154, [d, =, 6], 106.5, 0.5, []),
event(e155, [c, =, 6], 107, 0.75, []),
event(e156, [d, =, 6], 107.75, 0.25, []),
event(e157, [a, #, 5], 108, 1.0, []),
event(e158, [d, =, 6], 113.5, 0.5, []),
event(e159, [a, #, 5], 114, 0.75, []),
event(e160, [g, =, 5], 114.75, 0.25, []),
event(e161, [e, =, 5], 115, 0.5, []),
event(e162, [d, =, 6], 115.5, 0.5, []),
event(e163, [a, =, 5], 116, 0.75, []),
event(e164, [f, =, 5], 116.75, 0.25, []),
event(e165, [d, =, 5], 117, 0.5, []),
event(e166, [a, =, 5], 117.5, 0.5, []),
event(e167, [a, =, 5], 118, 0.5, []),
event(e168, [d, =, 6], 118.5, 0.5, []),
event(e169, [e, =, 6], 119, 0.5, []),
event(e170, [c, #, 6], 119.5, 0.5, []),
event(e171, [d, =, 6], 120, 1.0, []),
event(e172, [d, =, 6], 121.5, 0.5, []),
event(e173, [a, #, 5], 122, 0.75, []),
```

```
event(e174, [g, =, 5], 122.75, 0.25, []),
event(e175, [e, =, 5], 123, 0.5, []),
event(e176, [d, =, 6], 123.5, 0.5, []),
event(e177, [a, =, 5], 124, 0.75, []),
event(e178, [f, =, 5], 124.75, 0.25, []),
event(e179, [d, =, 5], 125, 0.5, []),
event(e180, [a, =, 5], 125.5, 0.5, []),
event(e181, [a, =, 5], 126, 0.5, []),
event(e182, [f, =, 6], 126.5, 0.5, []),
event(e183, [e, =, 6], 127, 0.5, []),
event(e184, [c, #, 6], 127.5, 0.5, []),
event(e185, [d, =, 6], 128, 1.0, []),
event(e186, [f, #, 6], 141.5, 0.5, []),
event(e187, [e, =, 6], 142, 0.5, []),
event(e188, [d, =, 6], 142.5, 0.5, []),
event(e189, [a, =, 5], 143, 0.5, []),
event(e190, [f, #, 5], 143.5, 0.5, []),
event(e191, [e, =, 5], 144, 0.75, []),
event(e192, [f, #, 5], 144.75, 0.25, []),
event(e193, [e, =, 5], 145, 0.5, []),
event(e194, [b, =, 5], 145.5, 0.5, []),
event(e195, [d, =, 6], 146, 0.75, []),
event(e196, [f, #, 5], 146.75, 0.25, []),
event(e197, [a, =, 5], 147, 0.25, []),
event(e198, [f, #, 5], 147.25, 0.25, []),
event(e199, [e, =, 5], 147.5, 0.25, []),
event(e200, [f, #, 5], 147.75, 0.25, []),
event(e201, [d, =, 5], 148, 1.0, []),
event(e202, [f, #, 6], 149.5, 0.5, []),
event(e203, [e, =, 6], 150, 0.5, []),
event(e204, [d, =, 6], 150.5, 0.5, []),
event(e205, [a, =, 5], 151, 0.5, []),
event(e206, [f, #, 5], 151.5, 0.5, []),
event(e207, [e, =, 5], 152, 0.75, []),
event(e208, [f, #, 5], 152.75, 0.25, []),
event(e209, [e, =, 5], 153, 0.5, []),
event(e210, [b, =, 5], 153.5, 0.5, []),
event(e211, [d, =, 6], 154, 0.75, []),
event(e212, [f, #, 5], 154.75, 0.25, []),
event(e213, [a, =, 5], 155, 0.25, []),
event(e214, [f, #, 5], 155.25, 0.25, []),
event(e215, [e, =, 5], 155.5, 0.25, []),
event(e216, [f, #, 5], 155.75, 0.25, []),
event(e217, [d, =, 5], 156, 1.0, []),
```

```
event(e218, [d, =, 5], 157.5, 0.5, []),
event(e219, [g, =, 5], 158, 0.5, []),
event(e220, [g, =, 5], 158.5, 0.5, []),
event(e221, [g, =, 5], 159, 0.5, []),
event(e222, [f, #, 5], 159.5, 0.25, []),
event(e223, [g, =, 5], 159.75, 0.25, []),
event(e224, [a, =, 5], 160, 0.75, []),
event(e225, [b, =, 5], 160.75, 0.25, []),
event(e226, [a, =, 5], 161, 0.5, []),
event(e227, [b, =, 5], 161.5, 0.5, []),
event(e228, [d, =, 6], 162, 0.25, []),
event(e229, [c, =, 6], 162.25, 0.25, []),
event(e230, [b, =, 5], 162.5, 0.5, []),
event(e231, [a, =, 5], 163, 0.5, []),
event(e232, [b, =, 5], 163.5, 0.25, []),
event(e233, [c, =, 6], 163.75, 0.25, []),
event(e234, [b, =, 5], 164, 1.0, []),
event(e235, [a, =, 5], 165.5, 0.5, []),
event(e236, [d, =, 6], 166, 0.5, []),
event(e237, [d, =, 6], 166.5, 0.5, []),
event(e238, [d, =, 6], 167, 0.5, []),
event(e239, [c, #, 6], 167.5, 0.25, []),
event(e240, [d, =, 6], 167.75, 0.25, []),
event(e241, [e, =, 6], 168, 0.75, []),
event(e242, [f, #, 6], 168.75, 0.25, []),
event(e243, [e, =, 6], 169, 0.5, []),
event(e244, [f, #, 6], 169.5, 0.5, []),
event(e245, [c, #, 6], 170, 0.5, []),
event(e246, [d, =, 6], 170.5, 0.5, []),
event(e247, [a, =, 5], 171, 0.75, []),
event(e248, [g, =, 5], 171.75, 0.25, []),
event(e249, [f, #, 5], 172, 1.0, []),
event(e250, [d, =, 6], 177.5, 0.5, []),
event(e251, [b, =, 5], 178, 0.75, []),
event(e252, [g, #, 5], 178.75, 0.25, []),
event(e253, [e, =, 5], 179, 0.5, []),
event(e254, [d, =, 6], 179.5, 0.5, []),
event(e255, [a, =, 5], 180, 0.75, []),
event(e256, [f, #, 5], 180.75, 0.25, []),
event(e257, [d, =, 5], 181, 0.5, []),
event(e258, [a, =, 5], 181.5, 0.5, []),
event(e259, [a, =, 5], 182, 0.5, []),
event(e260, [d, =, 6], 182.5, 0.5, []),
event(e261, [e, =, 6], 183, 0.75, []),
```

```
event(e262, [c, #, 6], 183.75, 0.25, []),
event(e263, [d, =, 6], 184, 1.0, []),
event(e264, [d, =, 6], 185.5, 0.5, []),
event(e265, [b, =, 5], 186, 0.75, []),
event(e266, [g, #, 5], 186.75, 0.25, []),
event(e267, [e, =, 5], 187, 0.5, []),
event(e268, [d, =, 6], 187.5, 0.5, []),
event(e269, [a, =, 5], 188, 0.75, []),
event(e270, [f, #, 5], 188.75, 0.25, []),
event(e271, [d, =, 5], 189, 0.5, []),
event(e272, [a, =, 5], 189.5, 0.5, []),
event(e273, [a, =, 5], 190, 0.5, []),
event(e274, [f, #, 6], 190.5, 0.5, []),
event(e275, [e, =, 6], 191, 0.5, []),
event(e276, [c, #, 6], 191.5, 0.5, []),
event(e277, [d, =, 6], 192, 1.0, []),
event(e278, [a, =, 5], 193.5, 0.5, []),
event(e279, [a, =, 5], 194, 0.5, []),
event(e280, [f, =, 6], 194.5, 0.5, []),
event(e281, [e, =, 6], 195, 0.5, []),
event(e282, [c, #, 6], 195.5, 0.5, []),
event(e283, [d, =, 6], 196, 1.0, []),

constituent(c001, stream, schubert, [e001, e002, e003, e004,
  e005, e006, e007, e008, e009, e010, e011, e012, e013, e014,
  e015, e016, e017, e018, e019, e020, e021, e022, e023, e024,
  e025, e026, e027, e028, e029, e030, e031, e032, e033, e034,
  e035, e036, e037, e038, e039, e040, e041, e042, e043, e044,
  e045, e046, e047, e048, e049, e050, e051, e052, e053, e054,
  e055, e056, e057, e058, e059, e060, e061, e062, e063, e064,
  e065, e066, e067, e068, e069, e070, e071, e072, e073, e074,
  e075, e076, e077, e078, e079, e080, e081, e082, e083, e084,
  e085, e086, e087, e088, e089, e090, e091, e092, e093, e094,
  e095, e096, e097, e098, e099, e100, e101, e102, e103, e104,
  e105, e106, e107, e108, e109, e110, e111, e112, e113, e114,
  e115, e116, e117, e118, e119, e120, e121, e122, e123, e124,
  e125, e126, e127, e128, e129, e130, e131, e132, e133, e134,
  e135, e136, e137, e138, e139, e140, e141, e142, e143, e144,
  e145, e146, e147, e148, e149, e150, e151, e152, e153, e154,
  e155, e156, e157, e158, e159, e160, e161, e162, e163, e164,
  e165, e166, e167, e168, e169, e170, e171, e172, e173, e174,
  e175, e176, e177, e178, e179, e180, e181, e182, e183, e184,
  e185, e186, e187, e188, e189, e190, e191, e192, e193, e194,
  e195, e196, e197, e198, e199, e200, e201, e202, e203, e204,
```

```
        e205, e206, e207, e208, e209, e210, e211, e212, e213, e214,
        e215, e216, e217, e218, e219, e220, e221, e222, e223, e224,
        e225, e226, e227, e228, e229, e230, e231, e232, e233, e234,
        e235, e236, e237, e238, e239, e240, e241, e242, e243, e244,
        e245, e246, e247, e248, e249, e250, e251, e252, e253, e254,
        e255, e256, e257, e258, e259, e260, e261, e262, e263, e264,
        e265, e266, e267, e268, e269, e270, e271, e272, e273, e274,
        e275, e276, e277, e278, e279, e280, e281, e282, e283])
]).
```

Appendix B

Grouping Analyses

B.1 Fauré's *Berceuse*

```
+ Events: e002 e003 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e003 e004 Rule(s): []
+ Events: e004 e005 Rule(s): []
+ Events: e005 e006 Rule(s): []
+ Events: e006 e007 Rule(s): [rule(gpr3a,4)]
+ Events: e007 e008 Rule(s): []
+ Events: e008 e009 Rule(s): []
+ Events: e009 e010 Rule(s): [rule(gpr2a,4)]
+ Events: e010 e011 Rule(s): []
+ Events: e011 e012 Rule(s): [rule(gpr2a_rest,7),rule(gpr3a,4)]
+ Events: e012 e013 Rule(s): []
+ Events: e013 e014 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e014 e015 Rule(s): []
+ Events: e015 e016 Rule(s): []
+ Events: e016 e017 Rule(s): [rule(gpr3b,6)]
+ Events: e017 e018 Rule(s): []
+ Events: e018 e019 Rule(s): [rule(gpr3a,4)]
+ Events: e019 e020 Rule(s): []
+ Events: e020 e021 Rule(s): [rule(gpr2a,4),rule(gpr3b,6)]
+ Events: e021 e022 Rule(s): []
+ Events: e022 e023 Rule(s): [rule(gpr2a_rest,7),rule(gpr3a,4),
    rule(gpr3b,6)]
+ Events: e023 e024 Rule(s): []
+ Events: e024 e025 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e025 e026 Rule(s): []
+ Events: e026 e027 Rule(s): []
```

```
+ Events: e027 e028 Rule(s): []
+ Events: e028 e029 Rule(s): [rule(gpr3a,4)]
+ Events: e029 e030 Rule(s): []
+ Events: e030 e031 Rule(s): []
+ Events: e031 e032 Rule(s): [rule(gpr2a,4)]
+ Events: e032 e033 Rule(s): []
+ Events: e033 e034 Rule(s): [rule(gpr2a_rest,7),rule(gpr3a,4),
    rule(gpr3b,6)]
+ Events: e034 e035 Rule(s): []
+ Events: e035 e036 Rule(s): [rule(gpr2b,5)]
+ Events: e036 e037 Rule(s): [rule(gpr3a,4)]
+ Events: e037 e038 Rule(s): []
+ Events: e038 e039 Rule(s): []
+ Events: e039 e040 Rule(s): []
+ Events: e040 e041 Rule(s): []
+ Events: e041 e042 Rule(s): []
+ Events: e042 e043 Rule(s): [rule(gpr2a,4),rule(gpr3b,6),
    rule(gpr3d,4)]
+ Events: e043 e044 Rule(s): []
+ Events: e044 e045 Rule(s): [rule(gpr2a,4),rule(gpr3b,6)]
+ Events: e045 e046 Rule(s): []
+ Events: e046 e047 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e047 e048 Rule(s): []
+ Events: e048 e049 Rule(s): []
+ Events: e049 e050 Rule(s): []
+ Events: e050 e051 Rule(s): [rule(gpr3a,4)]
+ Events: e051 e052 Rule(s): []
+ Events: e052 e053 Rule(s): []
+ Events: e053 e054 Rule(s): [rule(gpr2a,4)]
+ Events: e054 e055 Rule(s): []
+ Events: e055 e056 Rule(s): [rule(gpr2a_rest,7),rule(gpr3a,4)]
+ Events: e056 e057 Rule(s): []
+ Events: e057 e058 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e058 e059 Rule(s): []
+ Events: e059 e060 Rule(s): []
+ Events: e060 e061 Rule(s): []
+ Events: e061 e062 Rule(s): []
+ Events: e062 e063 Rule(s): [rule(gpr3a,4)]
+ Events: e063 e064 Rule(s): []
+ Events: e064 e065 Rule(s): [rule(gpr2a,4)]
+ Events: e065 e066 Rule(s): []
+ Events: e066 e067 Rule(s): [rule(gpr2a_rest,7),rule(gpr3a,4),
    rule(gpr3b,6)]
+ Events: e067 e068 Rule(s): []
```

```
+ Events: e068 e069 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e069 e070 Rule(s): []
+ Events: e070 e071 Rule(s): []
+ Events: e071 e072 Rule(s): [rule(gpr2a,4),rule(gpr3a,4)]
+ Events: e072 e073 Rule(s): []
+ Events: e073 e074 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e074 e075 Rule(s): []
+ Events: e075 e076 Rule(s): []
+ Events: e076 e077 Rule(s): [rule(gpr2a,4)]
+ Events: e077 e078 Rule(s): [rule(gpr3a,4),rule(gpr3b,6)]
+ Events: e078 e079 Rule(s): [rule(gpr2b,5)]
+ Events: e079 e080 Rule(s): []
+ Events: e080 e081 Rule(s): []
+ Events: e081 e082 Rule(s): [rule(gpr3a,4),rule(gpr3b,6),
    rule(gpr3d,4)]
+ Events: e082 e083 Rule(s): []
+ Events: e083 e084 Rule(s): []
+ Events: e084 e085 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e085 e086 Rule(s): []
+ Events: e086 e087 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e087 e088 Rule(s): []
+ Events: e088 e089 Rule(s): []
+ Events: e089 e090 Rule(s): []
+ Events: e090 e091 Rule(s): []
+ Events: e091 e092 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e092 e093 Rule(s): []
+ Events: e093 e094 Rule(s): [rule(gpr3a,4)]
+ Events: e094 e095 Rule(s): []
+ Events: e095 e096 Rule(s): []
+ Events: e096 e097 Rule(s): [rule(gpr2a,4)]
+ Events: e097 e098 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e098 e099 Rule(s): []
+ Events: e099 e100 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e100 e101 Rule(s): []
+ Events: e101 e102 Rule(s): [rule(gpr3a,4)]
+ Events: e102 e103 Rule(s): []
+ Events: e103 e104 Rule(s): []
+ Events: e104 e105 Rule(s): []
+ Events: e105 e106 Rule(s): [rule(gpr2a,4)]
+ Events: e106 e107 Rule(s): []
+ Events: e107 e108 Rule(s): []
```

```
+ Events: e108 e109 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e109 e110 Rule(s): []
+ Events: e110 e111 Rule(s): []
+ Events: e111 e112 Rule(s): []
+ Events: e112 e113 Rule(s): []
+ Events: e113 e114 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e114 e115 Rule(s): []
+ Events: e115 e116 Rule(s): [rule(gpr3a,4)]
+ Events: e116 e117 Rule(s): []
+ Events: e117 e118 Rule(s): []
+ Events: e118 e119 Rule(s): [rule(gpr2a,4)]
+ Events: e119 e120 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e120 e121 Rule(s): []
+ Events: e121 e122 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e122 e123 Rule(s): []
+ Events: e123 e124 Rule(s): [rule(gpr3a,4)]
+ Events: e124 e125 Rule(s): []
+ Events: e125 e126 Rule(s): []
+ Events: e126 e127 Rule(s): []
+ Events: e127 e128 Rule(s): []
+ Events: e128 e129 Rule(s): [rule(gpr2a,4),rule(gpr3b,6)]
+ Events: e129 e130 Rule(s): []
+ Events: e130 e131 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e131 e132 Rule(s): []
+ Events: e132 e133 Rule(s): []
+ Events: e133 e134 Rule(s): [rule(gpr2a,4)]
+ Events: e134 e135 Rule(s): []
+ Events: e135 e136 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e136 e137 Rule(s): []
+ Events: e137 e138 Rule(s): [rule(gpr3a,4)]
+ Events: e138 e139 Rule(s): []
+ Events: e139 e140 Rule(s): []
+ Events: e140 e141 Rule(s): [rule(gpr2a,4)]
+ Events: e141 e142 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e142 e143 Rule(s): []
+ Events: e143 e144 Rule(s): [rule(gpr2b,5),rule(gpr3b,6)]
+ Events: e144 e145 Rule(s): []
+ Events: e145 e146 Rule(s): []
+ Events: e146 e147 Rule(s): [rule(gpr3b,6)]
+ Events: e147 e148 Rule(s): []
+ Events: e148 e149 Rule(s): []
+ Events: e149 e150 Rule(s): [rule(gpr3a,4)]
```

```
+ Events: e150 e151 Rule(s): [rule(gpr2a,4),rule(gpr3b,6),
    rule(gpr3d,4)]
+ Events: e151 e152 Rule(s): []
+ Events: e152 e153 Rule(s): [rule(gpr3a,4)]
+ Events: e153 e154 Rule(s): []
+ Events: e154 e155 Rule(s): []
+ Events: e155 e156 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e156 e157 Rule(s): []
+ Events: e157 e158 Rule(s): [rule(gpr3a,4)]
+ Events: e158 e159 Rule(s): []
+ Events: e159 e160 Rule(s): []
+ Events: e160 e161 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e161 e162 Rule(s): []
+ Events: e162 e163 Rule(s): [rule(gpr3a,4)]
+ Events: e163 e164 Rule(s): []
+ Events: e164 e165 Rule(s): []
+ Events: e165 e166 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e166 e167 Rule(s): []
+ Events: e167 e168 Rule(s): [rule(gpr3a,4)]
+ Events: e168 e169 Rule(s): []
+ Events: e169 e170 Rule(s): []
+ Events: e170 e171 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e171 e172 Rule(s): []
+ Events: e172 e173 Rule(s): [rule(gpr3a,4)]
+ Events: e173 e174 Rule(s): []
+ Events: e174 e175 Rule(s): []
+ Events: e175 e176 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e176 e177 Rule(s): []
+ Events: e177 e178 Rule(s): [rule(gpr3a,4)]
+ Events: e178 e179 Rule(s): []
+ Events: e179 e180 Rule(s): []
+ Events: e180 e181 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e181 e182 Rule(s): []
+ Events: e182 e183 Rule(s): [rule(gpr3a,4)]
+ Events: e183 e184 Rule(s): []
+ Events: e184 e185 Rule(s): []
+ Events: e185 e186 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
```

```
+ Events: e186 e187 Rule(s): []
+ Events: e187 e188 Rule(s): [rule(gpr3a,4)]
+ Events: e188 e189 Rule(s): []
+ Events: e189 e190 Rule(s): []
+ Events: e190 e191 Rule(s): [rule(gpr2a,4),rule(gpr2b,5)]
+ Events: e191 e192 Rule(s): []
+ Events: e192 e193 Rule(s): [rule(gpr3a,4)]
+ Events: e193 e194 Rule(s): []
+ Events: e194 e195 Rule(s): []
+ Events: e195 e196 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e196 e197 Rule(s): []
+ Events: e197 e198 Rule(s): [rule(gpr3a,4)]
+ Events: e198 e199 Rule(s): []
+ Events: e199 e200 Rule(s): []
+ Events: e200 e201 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e201 e202 Rule(s): []
+ Events: e202 e203 Rule(s): [rule(gpr3a,4)]
+ Events: e203 e204 Rule(s): []
+ Events: e204 e205 Rule(s): []
+ Events: e205 e206 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e206 e207 Rule(s): []
+ Events: e207 e208 Rule(s): []
+ Events: e208 e209 Rule(s): []
+ Events: e209 e210 Rule(s): []
+ Events: e210 e211 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e211 e212 Rule(s): []
+ Events: e212 e213 Rule(s): []
+ Events: e213 e214 Rule(s): []
+ Events: e214 e215 Rule(s): []
+ Events: e215 e216 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e216 e217 Rule(s): []
+ Events: e217 e218 Rule(s): []
+ Events: e218 e219 Rule(s): []
+ Events: e219 e220 Rule(s): [rule(gpr3a,4)]
+ Events: e220 e221 Rule(s): []
+ Events: e221 e222 Rule(s): [rule(gpr2a_rest,7),rule(gpr3a,4),
    rule(gpr3b,6)]
+ Events: e222 e223 Rule(s): [rule(gpr2b,5)]
+ Events: e223 e224 Rule(s): []
+ Events: e224 e225 Rule(s): []
```

```
+ Events: e225 e226 Rule(s): []
+ Events: e226 e227 Rule(s): [rule(gpr3a,4)]
+ Events: e227 e228 Rule(s): []
+ Events: e228 e229 Rule(s): []
+ Events: e229 e230 Rule(s): []
+ Events: e230 e231 Rule(s): []
+ Events: e231 e232 Rule(s): [rule(gpr2a_rest,7),rule(gpr3a,4),
    rule(gpr3b,6)]
+ Events: e232 e233 Rule(s): []
+ Events: e233 e234 Rule(s): [rule(gpr3a,4)]
+ Events: e234 e235 Rule(s): []
+ Events: e235 e236 Rule(s): []
+ Events: e236 e237 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e237 e238 Rule(s): []
+ Events: e238 e239 Rule(s): [rule(gpr3a,4)]
+ Events: e239 e240 Rule(s): []
+ Events: e240 e241 Rule(s): []
+ Events: e241 e242 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e242 e243 Rule(s): []
+ Events: e243 e244 Rule(s): [rule(gpr3a,4)]
+ Events: e244 e245 Rule(s): []
+ Events: e245 e246 Rule(s): []
+ Events: e246 e247 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e247 e248 Rule(s): []
+ Events: e248 e249 Rule(s): [rule(gpr3a,4)]
+ Events: e249 e250 Rule(s): []
+ Events: e250 e251 Rule(s): []
+ Events: e251 e252 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e252 e253 Rule(s): []
+ Events: e253 e254 Rule(s): [rule(gpr3a,4)]
+ Events: e254 e255 Rule(s): []
+ Events: e255 e256 Rule(s): []
+ Events: e256 e257 Rule(s): [rule(gpr2a,4),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e257 e258 Rule(s): []
+ Events: e258 e259 Rule(s): [rule(gpr3a,4)]
+ Events: e259 e260 Rule(s): []
+ Events: e260 e261 Rule(s): [rule(gpr3d,4)]
```

B.2 Beethoven's *Auf dem Hügel sitz ich spähend*

```
+ Events: e002 e003 Rule(s): []
+ Events: e003 e004 Rule(s): [rule(gpr2b,5)]
+ Events: e004 e005 Rule(s): []
+ Events: e005 e006 Rule(s): []
+ Events: e006 e007 Rule(s): []
+ Events: e007 e008 Rule(s): [rule(gpr3a,4)]
+ Events: e008 e009 Rule(s): [rule(gpr2a_rest,7)]
+ Events: e009 e010 Rule(s): []
+ Events: e010 e011 Rule(s): []
+ Events: e011 e012 Rule(s): []
+ Events: e012 e013 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e013 e014 Rule(s): []
+ Events: e014 e015 Rule(s): [rule(gpr3a,4)]
+ Events: e015 e016 Rule(s): []
+ Events: e016 e017 Rule(s): [rule(gpr2b,5)]
+ Events: e017 e018 Rule(s): []
+ Events: e018 e019 Rule(s): []
+ Events: e019 e020 Rule(s): [rule(gpr3d,4)]
+ Events: e020 e021 Rule(s): []
+ Events: e021 e022 Rule(s): [rule(gpr3d,4)]
+ Events: e022 e023 Rule(s): []
+ Events: e023 e024 Rule(s): [rule(gpr3a,4),rule(gpr3b,6),
    rule(gpr3d,4)]
+ Events: e024 e025 Rule(s): []
+ Events: e025 e026 Rule(s): [rule(gpr3b,6)]
+ Events: e026 e027 Rule(s): [rule(gpr2b,5)]
+ Events: e027 e028 Rule(s): [rule(gpr2a,4),rule(gpr3b,6)]
+ Events: e028 e029 Rule(s): []
+ Events: e029 e030 Rule(s): []
+ Events: e030 e031 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e031 e032 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5),
    rule(gpr3b,6)]
+ Events: e032 e033 Rule(s): []
+ Events: e033 e034 Rule(s): []
+ Events: e034 e035 Rule(s): [rule(gpr2b,5)]
+ Events: e035 e036 Rule(s): []
+ Events: e036 e037 Rule(s): []
+ Events: e037 e038 Rule(s): []
+ Events: e038 e039 Rule(s): [rule(gpr3a,4)]
+ Events: e039 e040 Rule(s): [rule(gpr2a_rest,7)]
+ Events: e040 e041 Rule(s): []
+ Events: e041 e042 Rule(s): []
```

```
+ Events: e042 e043 Rule(s): []
+ Events: e043 e044 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e044 e045 Rule(s): []
+ Events: e045 e046 Rule(s): [rule(gpr3a,4)]
+ Events: e046 e047 Rule(s): []
+ Events: e047 e048 Rule(s): [rule(gpr2b,5)]
+ Events: e048 e049 Rule(s): []
+ Events: e049 e050 Rule(s): [rule(gpr3b,6)]
+ Events: e050 e051 Rule(s): [rule(gpr3d,4)]
+ Events: e051 e052 Rule(s): []
+ Events: e052 e053 Rule(s): [rule(gpr3d,4)]
+ Events: e053 e054 Rule(s): []
+ Events: e054 e055 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e055 e056 Rule(s): []
+ Events: e056 e057 Rule(s): [rule(gpr3b,6)]
+ Events: e057 e058 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e058 e059 Rule(s): [rule(gpr2a,4),rule(gpr3b,6)]
+ Events: e059 e060 Rule(s): []
+ Events: e060 e061 Rule(s): []
+ Events: e061 e062 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e062 e063 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5),
    rule(gpr3b,6)]
+ Events: e063 e064 Rule(s): []
+ Events: e064 e065 Rule(s): []
+ Events: e065 e066 Rule(s): [rule(gpr2b,5)]
+ Events: e066 e067 Rule(s): []
+ Events: e067 e068 Rule(s): []
+ Events: e068 e069 Rule(s): []
+ Events: e069 e070 Rule(s): [rule(gpr3a,4)]
+ Events: e070 e071 Rule(s): [rule(gpr2a_rest,7)]
+ Events: e071 e072 Rule(s): []
+ Events: e072 e073 Rule(s): [rule(gpr3b,6)]
+ Events: e073 e074 Rule(s): []
+ Events: e074 e075 Rule(s): []
+ Events: e075 e076 Rule(s): []
+ Events: e076 e077 Rule(s): [rule(gpr2a,4),rule(gpr2b,5)]
+ Events: e077 e078 Rule(s): [rule(gpr3b,6)]
+ Events: e078 e079 Rule(s): []
+ Events: e079 e080 Rule(s): [rule(gpr2b,5)]
+ Events: e080 e081 Rule(s): []
+ Events: e081 e082 Rule(s): []
+ Events: e082 e083 Rule(s): [rule(gpr3d,4)]
+ Events: e083 e084 Rule(s): []
+ Events: e084 e085 Rule(s): []
```

```
+ Events: e085 e086 Rule(s): []
+ Events: e086 e087 Rule(s): [rule(gpr2a_rest,7),rule(gpr3a,4)]
+ Events: e087 e088 Rule(s): []
+ Events: e088 e089 Rule(s): []
+ Events: e089 e090 Rule(s): [rule(gpr2b,5)]
+ Events: e090 e091 Rule(s): [rule(gpr2a,4)]
+ Events: e091 e092 Rule(s): []
+ Events: e092 e093 Rule(s): []
+ Events: e093 e094 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e094 e095 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5),
    rule(gpr3b,6)]
+ Events: e095 e096 Rule(s): []
+ Events: e096 e097 Rule(s): []
+ Events: e097 e098 Rule(s): [rule(gpr2b,5),rule(gpr3b,6)]
+ Events: e098 e099 Rule(s): []
+ Events: e099 e100 Rule(s): []
+ Events: e100 e101 Rule(s): []
+ Events: e101 e102 Rule(s): [rule(gpr3a,4)]
+ Events: e102 e103 Rule(s): [rule(gpr2a_rest,7)]
+ Events: e103 e104 Rule(s): []
+ Events: e104 e105 Rule(s): [rule(gpr3b,6)]
+ Events: e105 e106 Rule(s): []
+ Events: e106 e107 Rule(s): []
+ Events: e107 e108 Rule(s): [rule(gpr2b,5)]
+ Events: e108 e109 Rule(s): [rule(gpr2a,4)]
+ Events: e109 e110 Rule(s): []
+ Events: e110 e111 Rule(s): [rule(gpr3b,6)]
+ Events: e111 e112 Rule(s): [rule(gpr2b,5)]
+ Events: e112 e113 Rule(s): []
+ Events: e113 e114 Rule(s): []
+ Events: e114 e115 Rule(s): [rule(gpr3c,4)]
+ Events: e115 e116 Rule(s): [rule(gpr3a,4)]
+ Events: e116 e117 Rule(s): [rule(gpr2a_rest,7)]
+ Events: e117 e118 Rule(s): [rule(gpr3c,4)]
+ Events: e118 e119 Rule(s): []
+ Events: e119 e120 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e120 e121 Rule(s): []
+ Events: e121 e122 Rule(s): []
+ Events: e122 e123 Rule(s): [rule(gpr2b,5)]
+ Events: e123 e124 Rule(s): [rule(gpr2a,4)]
+ Events: e124 e125 Rule(s): []
+ Events: e125 e126 Rule(s): []
+ Events: e126 e127 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e127 e128 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5)]
```

```
+ Events: e128 e129 Rule(s): []
+ Events: e129 e130 Rule(s): []
+ Events: e130 e131 Rule(s): [rule(gpr2b,5)]
+ Events: e131 e132 Rule(s): []
+ Events: e132 e133 Rule(s): []
+ Events: e133 e134 Rule(s): []
+ Events: e134 e135 Rule(s): [rule(gpr3a,4)]
+ Events: e135 e136 Rule(s): [rule(gpr2a_rest,7)]
+ Events: e136 e137 Rule(s): []
+ Events: e137 e138 Rule(s): []
+ Events: e138 e139 Rule(s): []
+ Events: e139 e140 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e140 e141 Rule(s): [rule(gpr3b,6)]
+ Events: e141 e142 Rule(s): [rule(gpr3a,4)]
+ Events: e142 e143 Rule(s): []
+ Events: e143 e144 Rule(s): [rule(gpr2b,5)]
+ Events: e144 e145 Rule(s): []
+ Events: e145 e146 Rule(s): [rule(gpr2b,5)]
+ Events: e146 e147 Rule(s): [rule(gpr2a,4),rule(gpr3a,4)]
+ Events: e147 e148 Rule(s): []
+ Events: e148 e149 Rule(s): []
+ Events: e149 e150 Rule(s): [rule(gpr3d,4)]
+ Events: e150 e151 Rule(s): []
+ Events: e151 e152 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e152 e153 Rule(s): []
+ Events: e153 e154 Rule(s): []
+ Events: e154 e155 Rule(s): [rule(gpr2b,5)]
+ Events: e155 e156 Rule(s): [rule(gpr2a,4)]
+ Events: e156 e157 Rule(s): []
+ Events: e157 e158 Rule(s): []
```

B.3 Schubert's *Gute Nacht*

```
+ Events: e002 e003 Rule(s): []
+ Events: e003 e004 Rule(s): [rule(gpr3a,4)]
+ Events: e004 e005 Rule(s): []
+ Events: e005 e006 Rule(s): []
+ Events: e006 e007 Rule(s): [rule(gpr2b,5)]
+ Events: e007 e008 Rule(s): []
+ Events: e008 e009 Rule(s): [rule(gpr3a,4)]
+ Events: e009 e010 Rule(s): []
+ Events: e010 e011 Rule(s): [rule(gpr2b,5)]
+ Events: e011 e012 Rule(s): []
+ Events: e012 e013 Rule(s): []
+ Events: e013 e014 Rule(s): []
+ Events: e014 e015 Rule(s): []
+ Events: e015 e016 Rule(s): []
+ Events: e016 e017 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e017 e018 Rule(s): []
+ Events: e018 e019 Rule(s): []
+ Events: e019 e020 Rule(s): [rule(gpr3a,4)]
+ Events: e020 e021 Rule(s): []
+ Events: e021 e022 Rule(s): []
+ Events: e022 e023 Rule(s): [rule(gpr2b,5)]
+ Events: e023 e024 Rule(s): []
+ Events: e024 e025 Rule(s): [rule(gpr3a,4)]
+ Events: e025 e026 Rule(s): []
+ Events: e026 e027 Rule(s): [rule(gpr2b,5)]
+ Events: e027 e028 Rule(s): []
+ Events: e028 e029 Rule(s): []
+ Events: e029 e030 Rule(s): []
+ Events: e030 e031 Rule(s): []
+ Events: e031 e032 Rule(s): [rule(gpr3a,4)]
+ Events: e032 e033 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5)]
+ Events: e033 e034 Rule(s): [rule(gpr3a,4)]
+ Events: e034 e035 Rule(s): []
+ Events: e035 e036 Rule(s): []
+ Events: e036 e037 Rule(s): []
+ Events: e037 e038 Rule(s): []
+ Events: e038 e039 Rule(s): [rule(gpr2b,5)]
+ Events: e039 e040 Rule(s): []
+ Events: e040 e041 Rule(s): []
+ Events: e041 e042 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e042 e043 Rule(s): []
```

```
+ Events: e043 e044 Rule(s): [rule(gpr3d,4)]
+ Events: e044 e045 Rule(s): []
+ Events: e045 e046 Rule(s): [rule(gpr3d,4)]
+ Events: e046 e047 Rule(s): []
+ Events: e047 e048 Rule(s): []
+ Events: e048 e049 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5)]
+ Events: e049 e050 Rule(s): [rule(gpr3a,4)]
+ Events: e050 e051 Rule(s): []
+ Events: e051 e052 Rule(s): []
+ Events: e052 e053 Rule(s): []
+ Events: e053 e054 Rule(s): []
+ Events: e054 e055 Rule(s): [rule(gpr2b,5)]
+ Events: e055 e056 Rule(s): []
+ Events: e056 e057 Rule(s): []
+ Events: e057 e058 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e058 e059 Rule(s): []
+ Events: e059 e060 Rule(s): []
+ Events: e060 e061 Rule(s): []
+ Events: e061 e062 Rule(s): [rule(gpr2b,5)]
+ Events: e062 e063 Rule(s): []
+ Events: e063 e064 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5)]
+ Events: e064 e065 Rule(s): []
+ Events: e065 e066 Rule(s): [rule(gpr2b,5)]
+ Events: e066 e067 Rule(s): []
+ Events: e067 e068 Rule(s): [rule(gpr3a,4)]
+ Events: e068 e069 Rule(s): []
+ Events: e069 e070 Rule(s): [rule(gpr2b,5)]
+ Events: e070 e071 Rule(s): []
+ Events: e071 e072 Rule(s): [rule(gpr2b,5)]
+ Events: e072 e073 Rule(s): []
+ Events: e073 e074 Rule(s): [rule(gpr3d,4)]
+ Events: e074 e075 Rule(s): []
+ Events: e075 e076 Rule(s): []
+ Events: e076 e077 Rule(s): [rule(gpr3d,4)]
+ Events: e077 e078 Rule(s): []
+ Events: e078 e079 Rule(s): []
+ Events: e079 e080 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5)]
+ Events: e080 e081 Rule(s): [rule(gpr3a,4)]
+ Events: e081 e082 Rule(s): [rule(gpr2b,5)]
+ Events: e082 e083 Rule(s): []
+ Events: e083 e084 Rule(s): [rule(gpr3a,4)]
+ Events: e084 e085 Rule(s): []
+ Events: e085 e086 Rule(s): [rule(gpr2b,5)]
+ Events: e086 e087 Rule(s): []
```

```
+ Events: e087 e088 Rule(s): [rule(gpr2b,5)]
+ Events: e088 e089 Rule(s): []
+ Events: e089 e090 Rule(s): [rule(gpr3d,4)]
+ Events: e090 e091 Rule(s): []
+ Events: e091 e092 Rule(s): []
+ Events: e092 e093 Rule(s): []
+ Events: e093 e094 Rule(s): []
+ Events: e094 e095 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e095 e096 Rule(s): []
+ Events: e096 e097 Rule(s): []
+ Events: e097 e098 Rule(s): [rule(gpr3a,4)]
+ Events: e098 e099 Rule(s): []
+ Events: e099 e100 Rule(s): []
+ Events: e100 e101 Rule(s): [rule(gpr2b,5)]
+ Events: e101 e102 Rule(s): []
+ Events: e102 e103 Rule(s): []
+ Events: e103 e104 Rule(s): [rule(gpr3a,4)]
+ Events: e104 e105 Rule(s): [rule(gpr2b,5)]
+ Events: e105 e106 Rule(s): []
+ Events: e106 e107 Rule(s): []
+ Events: e107 e108 Rule(s): []
+ Events: e108 e109 Rule(s): []
+ Events: e109 e110 Rule(s): []
+ Events: e110 e111 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e111 e112 Rule(s): []
+ Events: e112 e113 Rule(s): []
+ Events: e113 e114 Rule(s): [rule(gpr3a,4)]
+ Events: e114 e115 Rule(s): []
+ Events: e115 e116 Rule(s): []
+ Events: e116 e117 Rule(s): [rule(gpr2b,5)]
+ Events: e117 e118 Rule(s): []
+ Events: e118 e119 Rule(s): []
+ Events: e119 e120 Rule(s): [rule(gpr3a,4)]
+ Events: e120 e121 Rule(s): [rule(gpr2b,5)]
+ Events: e121 e122 Rule(s): []
+ Events: e122 e123 Rule(s): []
+ Events: e123 e124 Rule(s): []
+ Events: e124 e125 Rule(s): []
+ Events: e125 e126 Rule(s): []
+ Events: e126 e127 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e127 e128 Rule(s): []
```

```
+ Events: e128 e129 Rule(s): []
+ Events: e129 e130 Rule(s): []
+ Events: e130 e131 Rule(s): []
+ Events: e131 e132 Rule(s): []
+ Events: e132 e133 Rule(s): [rule(gpr2b,5)]
+ Events: e133 e134 Rule(s): []
+ Events: e134 e135 Rule(s): []
+ Events: e135 e136 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e136 e137 Rule(s): []
+ Events: e137 e138 Rule(s): [rule(gpr3d,4)]
+ Events: e138 e139 Rule(s): []
+ Events: e139 e140 Rule(s): [rule(gpr3d,4)]
+ Events: e140 e141 Rule(s): []
+ Events: e141 e142 Rule(s): []
+ Events: e142 e143 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5)]
+ Events: e143 e144 Rule(s): [rule(gpr3a,4)]
+ Events: e144 e145 Rule(s): []
+ Events: e145 e146 Rule(s): []
+ Events: e146 e147 Rule(s): []
+ Events: e147 e148 Rule(s): []
+ Events: e148 e149 Rule(s): [rule(gpr2b,5)]
+ Events: e149 e150 Rule(s): []
+ Events: e150 e151 Rule(s): []
+ Events: e151 e152 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e152 e153 Rule(s): []
+ Events: e153 e154 Rule(s): []
+ Events: e154 e155 Rule(s): []
+ Events: e155 e156 Rule(s): [rule(gpr2b,5)]
+ Events: e156 e157 Rule(s): []
+ Events: e157 e158 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5)]
+ Events: e158 e159 Rule(s): []
+ Events: e159 e160 Rule(s): [rule(gpr2b,5)]
+ Events: e160 e161 Rule(s): []
+ Events: e161 e162 Rule(s): [rule(gpr3a,4)]
+ Events: e162 e163 Rule(s): []
+ Events: e163 e164 Rule(s): [rule(gpr2b,5)]
+ Events: e164 e165 Rule(s): []
+ Events: e165 e166 Rule(s): [rule(gpr3a,4)]
+ Events: e166 e167 Rule(s): []
+ Events: e167 e168 Rule(s): [rule(gpr3a,4)]
+ Events: e168 e169 Rule(s): []
+ Events: e169 e170 Rule(s): [rule(gpr3a,4)]
+ Events: e170 e171 Rule(s): []
+ Events: e171 e172 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5)]
```

```
+ Events: e172 e173 Rule(s): [rule(gpr3a,4)]
+ Events: e173 e174 Rule(s): [rule(gpr2b,5)]
+ Events: e174 e175 Rule(s): []
+ Events: e175 e176 Rule(s): [rule(gpr3a,4)]
+ Events: e176 e177 Rule(s): []
+ Events: e177 e178 Rule(s): [rule(gpr2b,5)]
+ Events: e178 e179 Rule(s): []
+ Events: e179 e180 Rule(s): [rule(gpr3a,4)]
+ Events: e180 e181 Rule(s): []
+ Events: e181 e182 Rule(s): [rule(gpr3a,4)]
+ Events: e182 e183 Rule(s): []
+ Events: e183 e184 Rule(s): [rule(gpr3a,4)]
+ Events: e184 e185 Rule(s): []
+ Events: e185 e186 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e186 e187 Rule(s): []
+ Events: e187 e188 Rule(s): []
+ Events: e188 e189 Rule(s): [rule(gpr3a,4)]
+ Events: e189 e190 Rule(s): []
+ Events: e190 e191 Rule(s): []
+ Events: e191 e192 Rule(s): [rule(gpr2b,5)]
+ Events: e192 e193 Rule(s): []
+ Events: e193 e194 Rule(s): [rule(gpr3a,4)]
+ Events: e194 e195 Rule(s): []
+ Events: e195 e196 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e196 e197 Rule(s): []
+ Events: e197 e198 Rule(s): []
+ Events: e198 e199 Rule(s): []
+ Events: e199 e200 Rule(s): []
+ Events: e200 e201 Rule(s): []
+ Events: e201 e202 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e202 e203 Rule(s): []
+ Events: e203 e204 Rule(s): []
+ Events: e204 e205 Rule(s): [rule(gpr3a,4)]
+ Events: e205 e206 Rule(s): []
+ Events: e206 e207 Rule(s): []
+ Events: e207 e208 Rule(s): [rule(gpr2b,5)]
+ Events: e208 e209 Rule(s): []
+ Events: e209 e210 Rule(s): [rule(gpr3a,4)]
+ Events: e210 e211 Rule(s): []
+ Events: e211 e212 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e212 e213 Rule(s): []
+ Events: e213 e214 Rule(s): []
```

```
+ Events: e214 e215 Rule(s): []
+ Events: e215 e216 Rule(s): []
+ Events: e216 e217 Rule(s): [rule(gpr3a,4)]
+ Events: e217 e218 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5)]
+ Events: e218 e219 Rule(s): [rule(gpr3a,4)]
+ Events: e219 e220 Rule(s): []
+ Events: e220 e221 Rule(s): []
+ Events: e221 e222 Rule(s): [rule(gpr3d,4)]
+ Events: e222 e223 Rule(s): []
+ Events: e223 e224 Rule(s): []
+ Events: e224 e225 Rule(s): [rule(gpr2b,5)]
+ Events: e225 e226 Rule(s): []
+ Events: e226 e227 Rule(s): []
+ Events: e227 e228 Rule(s): [rule(gpr3a,4),rule(gpr3d,4)]
+ Events: e228 e229 Rule(s): []
+ Events: e229 e230 Rule(s): [rule(gpr3d,4)]
+ Events: e230 e231 Rule(s): []
+ Events: e231 e232 Rule(s): [rule(gpr3d,4)]
+ Events: e232 e233 Rule(s): []
+ Events: e233 e234 Rule(s): []
+ Events: e234 e235 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5)]
+ Events: e235 e236 Rule(s): [rule(gpr3a,4)]
+ Events: e236 e237 Rule(s): []
+ Events: e237 e238 Rule(s): []
+ Events: e238 e239 Rule(s): [rule(gpr3d,4)]
+ Events: e239 e240 Rule(s): []
+ Events: e240 e241 Rule(s): []
+ Events: e241 e242 Rule(s): [rule(gpr2b,5)]
+ Events: e242 e243 Rule(s): []
+ Events: e243 e244 Rule(s): []
+ Events: e244 e245 Rule(s): [rule(gpr3a,4)]
+ Events: e245 e246 Rule(s): []
+ Events: e246 e247 Rule(s): [rule(gpr3a,4)]
+ Events: e247 e248 Rule(s): [rule(gpr2b,5)]
+ Events: e248 e249 Rule(s): []
+ Events: e249 e250 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e250 e251 Rule(s): []
+ Events: e251 e252 Rule(s): [rule(gpr2b,5)]
+ Events: e252 e253 Rule(s): []
+ Events: e253 e254 Rule(s): [rule(gpr3a,4)]
+ Events: e254 e255 Rule(s): []
+ Events: e255 e256 Rule(s): [rule(gpr2b,5)]
+ Events: e256 e257 Rule(s): []
```

```
+ Events: e257 e258 Rule(s): [rule(gpr3a,4)]
+ Events: e258 e259 Rule(s): []
+ Events: e259 e260 Rule(s): [rule(gpr3a,4)]
+ Events: e260 e261 Rule(s): []
+ Events: e261 e262 Rule(s): [rule(gpr2b,5),rule(gpr3a,4)]
+ Events: e262 e263 Rule(s): []
+ Events: e263 e264 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5)]
+ Events: e264 e265 Rule(s): []
+ Events: e265 e266 Rule(s): [rule(gpr2b,5)]
+ Events: e266 e267 Rule(s): []
+ Events: e267 e268 Rule(s): [rule(gpr3a,4)]
+ Events: e268 e269 Rule(s): []
+ Events: e269 e270 Rule(s): [rule(gpr2b,5)]
+ Events: e270 e271 Rule(s): []
+ Events: e271 e272 Rule(s): [rule(gpr3a,4)]
+ Events: e272 e273 Rule(s): []
+ Events: e273 e274 Rule(s): [rule(gpr3a,4)]
+ Events: e274 e275 Rule(s): []
+ Events: e275 e276 Rule(s): [rule(gpr3a,4)]
+ Events: e276 e277 Rule(s): []
+ Events: e277 e278 Rule(s): [rule(gpr2a_rest,7),rule(gpr2b,5),
    rule(gpr3a,4)]
+ Events: e278 e279 Rule(s): []
+ Events: e279 e280 Rule(s): [rule(gpr3a,4)]
+ Events: e280 e281 Rule(s): []
+ Events: e281 e282 Rule(s): [rule(gpr3a,4)]
```

Appendix C

Partial Autocorrelation

The following is a brief discussion of how the partial autocorrelation estimates for a time series can be calculated. A more detailed discussion of this process is contained in Box and Jenkins (1976).

If a time-series is autoregressive of order k, then it can be represented by:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_k y_{t-k} + a_t \tag{C.1}$$

From this we can derive the equations of the autocorrelations ρ_j:

$$\rho_j = \phi_1 \rho_{j-1} + \phi_2 \rho_{j-2} + \ldots + \phi_k \rho_{j-k}, \quad j = 1, 2, \ldots k \tag{C.2}$$

More generally, the Yule-Walker equations presented below can be obtained (where ϕ_{kj} is the ϕ_j coefficient in an autoregressive model of order k):

$$\begin{aligned}
\rho_1 &= \phi_{k1} + \phi_{k2}\rho_1 + \ldots + \phi_{kk}\rho_{k-1} \\
\rho_2 &= \phi_{k1}\rho_1 + \phi_{k2} + \ldots + \phi_{kk}\rho_{k-2} \\
&\vdots \\
\rho_k &= \phi_{k1}\rho_{k-1} + \phi_{k2}\rho_{k-2} + \ldots + \phi_{kk}
\end{aligned} \tag{C.3}$$

Which can be represented in matrix form as follows:

$$\begin{pmatrix} 1 & \rho_1 & \rho_2 & \ldots & \rho_{k-1} \\ \rho_1 & 1 & \rho_1 & \ldots & \rho_{k-2} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \rho_{k-1} & \rho_{k-2} & \ldots & \rho_1 & 1 \end{pmatrix} \begin{pmatrix} \phi_{k1} \\ \phi_{k2} \\ \ldots \\ \phi_{kk} \end{pmatrix} = \begin{pmatrix} \rho_1 \\ \rho_2 \\ \ldots \\ \rho_k \end{pmatrix} \tag{C.4}$$

Replacing the autocorrelation functions ρ_j with their estimates (r_j) derived from the time-series data, the estimated values of ϕ (denoted as $\hat{\phi}$) can be derived (where $\hat{\phi}_{kk}$ is the partial autocorrelation estimate for lag k).

For example, given an autoregressive model of order 2, the following sets of equations are obtained:

$$\begin{pmatrix} 1 \end{pmatrix} \begin{pmatrix} \phi_{11} \end{pmatrix} = \begin{pmatrix} \rho_1 \end{pmatrix} \tag{C.5}$$

$$\begin{pmatrix} 1 & \rho_1 \\ \rho_1 & 1 \end{pmatrix} \begin{pmatrix} \phi_{21} \\ \phi_{22} \end{pmatrix} = \begin{pmatrix} \rho_1 \\ \rho_2 \end{pmatrix} \tag{C.6}$$

From which, by replacing the theoretical autocorrelations ρ_k by the estimated autocorrelations r_k, the estimated partial autocorrelations $\hat{\phi}_{11}$ and $\hat{\phi}_{22}$ can be obtained as follows. First, the matrix multiplications are performed to provide the following solutions:

$$\hat{\phi}_{11} = r_1 \tag{C.7}$$

$$\hat{\phi}_{21} + r_1\hat{\phi}_{22} = r_1 \tag{C.8}$$

$$r_1\hat{\phi}_{21} + \hat{\phi}_{22} = r_2 \tag{C.9}$$

Trivially, from Equation C.7, $\hat{\phi}_{11} = r_1$. To calculate $\hat{\phi}_{22}$ equations C.8 and C.9 can be rearranged as follows:

$$\hat{\phi}_{21} = r_1 - r_1\hat{\phi}_{22} \tag{C.10}$$

$$\hat{\phi}_{21} = \frac{r_2 - \hat{\phi}_{22}}{r_1} \tag{C.11}$$

Now, by equating C.10 and C.11:

$$r_1 - r_1\hat{\phi}_{22} = \frac{r_2 - \hat{\phi}_{22}}{r_1} \tag{C.12}$$

$$r_1^2 - r_1^2\hat{\phi}_{22} = r_2 - \hat{\phi}_{22} \tag{C.13}$$

$$\hat{\phi}_{22}(1 - r_1^2) = r_2 - r_1^2 \tag{C.14}$$

$$\hat{\phi}_{22} = \frac{r_2 - r_1^2}{1 - r_1^2} \tag{C.15}$$

The resulting partial autocorrelation estimates $\hat{\phi}_{11}$ and $\hat{\phi}_{22}$ are now expressed entirely in terms of the autocorrelation estimates r_1 and r_2. A similar process can be followed for values of $k > 2$.

The Levinson-Durbin recursive algorithm provides an efficient method to calculate the $\hat{\phi}_{kk}$ values and is presented below:

$$\hat{\phi}_{11} = r_1 \tag{C.16}$$

$$\hat{\phi}_{22} = \frac{\left(r_2 - r_1^2\right)}{\left(1 - r_1^2\right)} \tag{C.17}$$

$$\hat{\phi}_{kj} = \hat{\phi}_{k-1,\,j} - \hat{\phi}_{kk}\hat{\phi}_{k-1,k-j} \tag{C.18}$$

$$\hat{\phi}_{kk} = \frac{\left(r_k - \sum_{j=1}^{k-1}\hat{\phi}_{k-1,\,j}r_{k-j}\right)}{\left(1 - \sum_{j=1}^{k-1}\hat{\phi}_{k-1,\,j}r_j\right)} \tag{C.19}$$

Appendix D

Published Papers

The following papers have been published as a result of research presented in this dissertation.

Ben Curry and Geraint A. Wiggins. A new approach to cooperative performance: A preliminary experiment. *International Journal of Computing Anticipatory Systems*, 4:163–178, 1999. . . . . pp. 312–328

Ben Curry, Geraint A. Wiggins, and Gillian Hayes. Representing trees with constraints. In J. Lloyd *et al.*, editors, *Proceedings of the First International Conference on Computational Logic*, volume 1861 of *LNAI*, pages 315–325. Springer Verlag, 2000. . . . . pp. 329–339

A New Approach to Cooperative Performance: A Preliminary Experiment

Ben Curry and Geraint Wiggins
School of Artificial Intelligence
Division of Informatics, University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, Scotland
Email: {benc,geraint}@dai.ed.ac.uk
Fax: +44-131-650-6516

Abstract

This paper describes a preliminary investigation into the relationship between the high-level structure of music and expressive performance. The experiment aims to determine whether it is possible in principle to develop a system which uses both the structure of a piece and real-time performance data from a human musician to provide an expressive performance.

Keywords: Expression, Performance, Structure, GTTM, Music

1 Introduction

The purpose of this experiment is to investigate the relationship between the high-level structure of a piece of music and the expressive performance of that piece and whether it can be used in automated accompaniment.

This investigation forms part of our research into creating a cooperative performer that, instead of merely tracking a human performer and playing along, has some notion of the structure of a piece and will adjust its performance both according to this structure and the human's performance.

The work presented below has been done with the intention of building a computer system to do the same task. We have been adopting the role of the system whilst we performed this preliminary experiment. The long term aim of our research is to construct a system that will perform all these tasks and then be capable of using the information gathered to produce an expressive performance in the manner described above.

The structure of this paper is as follows: first, we discuss some work related to this experiment. Then we describe the various decisions that were made when we designed the experiment. A description of how we analysed the raw data is given and then we go on to present the results. Finally, we present our conclusions from this preliminary experiment.

2 Related Work

2.1 Score Tracking

Score tracking tries to match events in the performance with events in the score. Roger Dannenberg has made significant contributions to this field (e.g. Dannenberg and Mukaino, 1988 ; Dannenberg, 1993) and has recently developed a system along with Lorin Grubb (1997) that tracks and accompanies a vocal performer using statistical methods. Another recent body of work by Desain et al. (1997) has focussed on using properties of music, such as the relatively strict order of events, to assist with the matching process. None of these techniques use any information about the structure of the music they are matching.

2.2 Expressive Performance

Generating expressive performances is another important field of research. We will mention three recent noteworthy attempts to solve this problem. Arcos et al. (1997) used case based reasoning to generate expressive performances from mechanical ones. The SaxEx system, which they have developed, takes a mechanical performance of a piece of music along with a MIDI file containing the harmonic and melodic information and attempts to find a similar phrase amongst its stored cases. If it finds a match, it applies the same deviations which were used in an expressive performance of the stored case to this new phrase.

Friberg et al. (1997) use a technique which they call “Analysis by Synthesis” to create a set of rules to perform musical “punctuation”. This technique involves giving the system a rule which it is believed is used in performing this punctuation task and then gradually improving the rule through a process of using, evaluating and then adjusting the rule.

Gerhard Widmer (1995) used machine learning techniques to produce a set of rules which encapsulated the expressive deviations introduced by a performer. He presents his system with a score and an expressive performance of that score and lets the system discover the rules behind that expressive performance.

2.3 Musical Understanding

Human listeners are capable of deriving more information about a piece of music than is present in the score. They attribute various different structures and relationships to the music which are rarely explicit in the score. Here, we briefly mention two theories and then describe in more detail a third, which we use in the current research.

Eugene Narmour’s (1992) theory of analysis is based upon expectation. His theory attempts to encompass “the specific, note-to-note principles by which listeners perceive, structure and comprehend the vast world of melody”. The model defines style in terms of bottom-up shapes and top-down structures. The bottom-up shapes are based upon the

perception of events and the top-down structures are based upon "complex hierarchical interrelations".

Recent work by Emilios Cambouropoulos (1998) has highlighted the need for a theory which is based upon "general cognitive and logical principles". His theory attempts to derive a high-level structure of a musical piece independent of musical style or idiom. Although attractive because it is a very formal theory, it is still in the process of being implemented and so cannot be used for our research.

The authors of The Generative Theory of Tonal Music (GTTM) state its goal to be a "*formal description of the musical intuitions of a listener who is experienced in a musical idiom*" (Lerdahl and Jackendoff, 1983). GTTM attempts to formalise the process of how an "experienced listener" perceives the structure of the piece being heard. The theory itself is split up into four sections which each have three different types of rules within them. The three different types of rules are:

Well-formedness Rules which state what sort of structural descriptions are possible.

Preference Rules which try to select from the possible structures the ones that correspond to what an experienced listener would hear.

Transformational Rules that allow certain "distortions" of the strict structures prescribed by the well-formedness rules.

The theory is broadly divided into two groups of rules. The first describes rhythmic structure which encompasses both the grouping structure and the metrical structure of a piece. The second group is based upon the notion of reduction in which some musical events are structurally less important than others. These three types of rule are stated for each of the four different, but related, structural properties of a piece which are dealt with by GTTM.

Grouping Structure is the segmentation of musical events into groups of similar or related events. These rules try to encapsulate the notion of "chunking" in which a listener groups certain events together whilst hearing the piece.

Metrical Structure models the relative strength and weakness of events at various levels in a metrical hierarchy. It captures the notion of strong and weak beats.

Time-span Reduction identifies pitch events which are perceived as being of greater structural importance at various levels.

Prolongational Reduction identifies events which represent the harmonic movement of the piece. The prolongational reduction deals with issues such as tension, relaxation, continuity and progression.

Although the authors attempt to be thorough and formal throughout the theory, they do not resolve much of the ambiguity that exists through the application of the preference

rules. There is little or no ranking of these rules to say which should be preferred over others and this detracts from what was presented as a formal theory.

Criticisms aside, GTTM does provide a very useful and important framework from which we can begin to build autonomous systems that should be able to analyse a piece and construct the appropriate high-level structures, even if multiple possible solutions may be generated.

The aim of this preliminary experiment is to explore the relationship between GTTM and an expressive performance of a piece. If there does appear to be such a relationship, further, more extensive, work is needed to try and discover its mathematical properties. In the meantime, this experiment is expected merely to point towards the existence of a relationship and not expected to provide any sound understanding of what exactly the relationship is.

3 Design

The first stage in the experiment was selecting an appropriate piece of music for analysis. We needed a piece that was both relatively simple but was not too trivial. The piece needed to be a duet which could be performed on a MIDI enabled instrument.

We chose Gabriel Fauré's *Dolly Suite: Berceuse* (1894). It is a relatively slow piece for two pianos which has two mostly distinct parts. The first part is largely responsible for the melody and the second part provides the bass and the harmony. The piece is intended to be performed on one piano with the two parts being played mostly in non-intersecting ranges of the keyboard.

Two pianists were chosen to perform the piece: one a professional pianist with more than ten years of experience, the other an experienced amateur pianist.

After a brief period of rehearsal to get used to the piece and the MIDI enabled piano, three separate performances were recorded. The performances were recorded using an Apple Macintosh SE/30 with EZVision software recording the MIDI data.

4 Analysis

The MIDI data files were transferred to a PC and the two parts were separated using Evolution Audio Lite by Evolution Electronics (UK). A C program modified from Watkinson (1997) was used to extract the relevant information from the midi files (i.e., pitch, onset time, duration and velocity).

For the purposes of this preliminary experiment we chose to analyse just the first 22 bars of the piece. This contained five complete occurrences of the 4 bar theme with some repetition and variation and with the first two bars being silent.

In the case of the first voice, the melody is played by both hands with the notes an octave apart. This meant that we could average both the velocity and timing information of

the notes which are played simultaneously. In most cases there was no difference between the onset times or the velocity, but if there was a difference it tended to be very small.

The next step was to calculate the average velocity and the deviations from that during the performance. This was a simple calculation which involved adding all the velocity information together and dividing it by the number of note positions. The relative deviation for each note position was then calculated by subtracting the average from the actual velocity at that point.

The timing information is harder to derive. To investigate the timing information, we need to take into account the fact that we are most interested in the onset time relative to the previous note rather than the absolute onset time within the piece. If we chose the latter option we would find that an initial difference in timing would then put all of the subsequent notes out of synch with our timing scale and give us distorted information. Instead we measured the deviation in relation to when the previous note was performed. This meant that we effectively had to re-calibrate the system at each note.

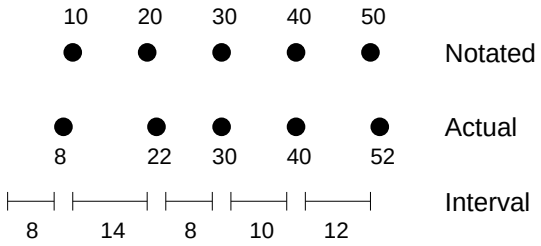


Figure 1: Illustration of re-calibration in action

Shift	0	-2	2	0	0
Notated Time	10	20	30	40	50
Calibrated Time	10	18	32	40	50
Actual Time	8	22	30	40	52
Difference	-2	+4	-2	0	2

Table 1: Re-calibration in action

If we take the example of the performance given in Figure 1 the first row of dots shows when the events should ideally occur given the information from the score. The second row of dots show when the notes were performed. Finally, the interval between the performed events is shown in the last row. The algorithm works as follows (using data from Table 1), we initially set the *shift* to be 0. We then calculate the difference between the time of the event that was performed and the time that was prescribed in the score. In the case of the first event the difference is -2 meaning that the event was performed 2 units of time early. We need to use this difference to calculate when we should expect the next event.

Since the first event was played 2 units early, we should then expect the second event to be performed with the same deviation. So we add -2 to the *notated* time of the next event to get our *calibrated* time. Now when we calculate the difference between the notated time and the performed time we get the correct deviation of +4.

Once we had timing and velocity data for the three performances, we averaged them to try and reduce some of the noise which might be present in the performances. The averaging was done using the deviations not the original data.

In order to investigate the data fully we investigated not only the overall timing and velocity deviations for the first 22 bars as a whole, but also what happened when we took each phrase (4 bars) in turn.

5 Results

The following section is divided into two parts; the first part describes some of the results of the GTTM analysis of the piece, the second presents the data obtained from the experiment and highlights some significant relationships.

5.1 GTTM Analysis

Shown in Table 2 are some of the more interesting features of the piece that helped to decide on the shape of the final structures selected. The first stage in the analysis is to examine the grouping and metrical structures of the piece. Figure 2 shows how these look for the first 4 bars of interest (the first 2 bars are silent). We can see that the tie in bar 4 causes the segmentation at the lowest level. The metrical structure, i.e. the strong and weak beats, were chosen partly due to the relatively long notes that occur at those points and partly to the location of the dynamics. It is this grouping and metrical structure that appears in most of the subsequent phrases.

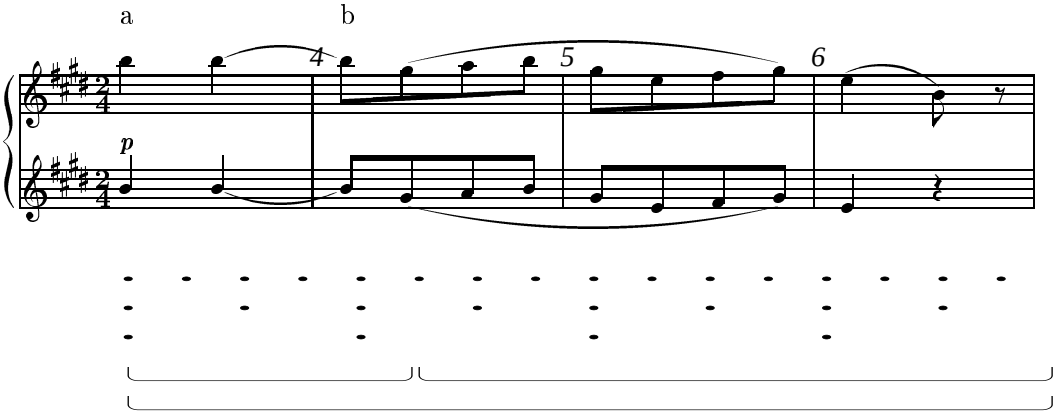


Figure 2: Metrical and Grouping structure for bars 3–6

Point of interest	Location (Bar:Beat)	Indicated by (Figure:Symbol)
Piano	3:1	Fig. 10:*a*
Tie across barline	4:1, 8:1	Fig. 10:*b*,*c*
Crescendo	9:1,2	Fig. 10:*d*
Diminuendo	10:1,2	Fig. 10:*e*
Piano	11:1	Fig. 12:*f*
Modulation	11:1	Fig. 12:*g*
Tie across barline	12:1	Fig. 12:*h*
Crescendo	15:1 – 17:2	Fig. 12:*i*
Tie across barline	16:1	Fig. 12:*j*
Diminuendo	18:1,2	Fig. 12:*k*
Piano	19:1	Fig. 14:*l*
Modulation	19:1	Fig. 14:*m*
Tie across barline	20:1	Fig. 14:*n*

Table 2: Points of interest in the piece

Bars 7–10 (see Figure 3) provide a slightly different grouping due to the modulation that begins in the middle of bar 9. This causes the extra division within the grouping structure which appears at the lowest level. One final point of interest occurs at the cadence at the end of bar 18 and beginning of bar 19. This cadence creates what is referred to as an “overlap” in the grouping structure which means that the first event of bar 19 is shared both with the previous phrase (to provide the cadence) and as the start of the next phrase.

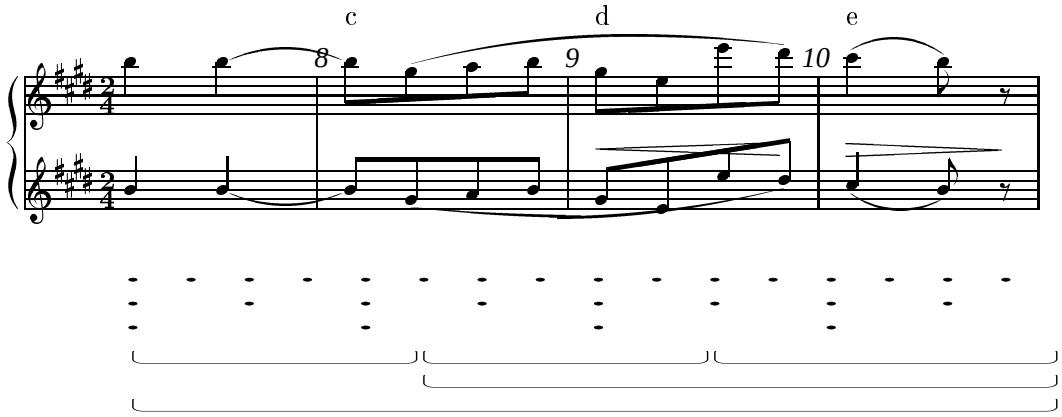


Figure 3: Metrical and Grouping structure for bars 7–10

The next two analyses that were performed are the time-span and prolongational reductions. The time-span reduction is strongly influenced by the metrical and grouping

structures that occur at that point in the piece. Figure 4 shows the shape of the time-span analysis that occurs for the first, third and final phrases.

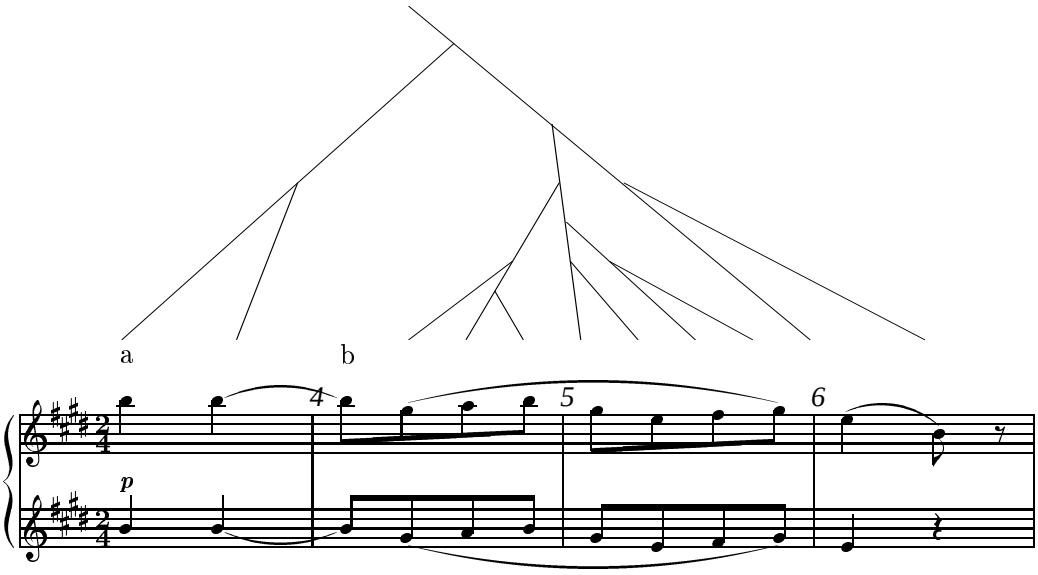


Figure 4: Time-span reduction for bars 3–6

Since the grouping structure is different for the second phrase (due to the beginning modulation) the resulting time-span analysis is also affected (see Figure 5). We can see that the last two notes in the third bar of the phrase (i.e., bar 9) have moved from being elaborations of the first note in that bar to elaborations of the first note in the last bar.

The fourth phrase provides a more complicated time-span analysis due to the cadence which links the fourth and fifth phrases (see Figure 6). This creates a significantly different structure for the final three events (18:1,2 & 19:1). Whereas in the previous time-span reductions the penultimate note of the phrase was highlighted as the most significant, in this case the C# at the end of bar 18 is highlighted.

The prolongational reduction in Figure 7 shows the shape of the analysis that covers all but the fourth phrase. The structure indicates that the phrase moves towards resolution at the penultimate note in the phrase. We can also see that the notes at 4:2,3,4 act to create a minor amount of tension that is resolved at 5:1, similarly notes 5:2,3,4 lead to the stable note at 6:1.

The structure shown in Figure 8 shows how the cadence affects the location of the focus. Again the focus has moved to the final event in the phrase and the event before has become less important.

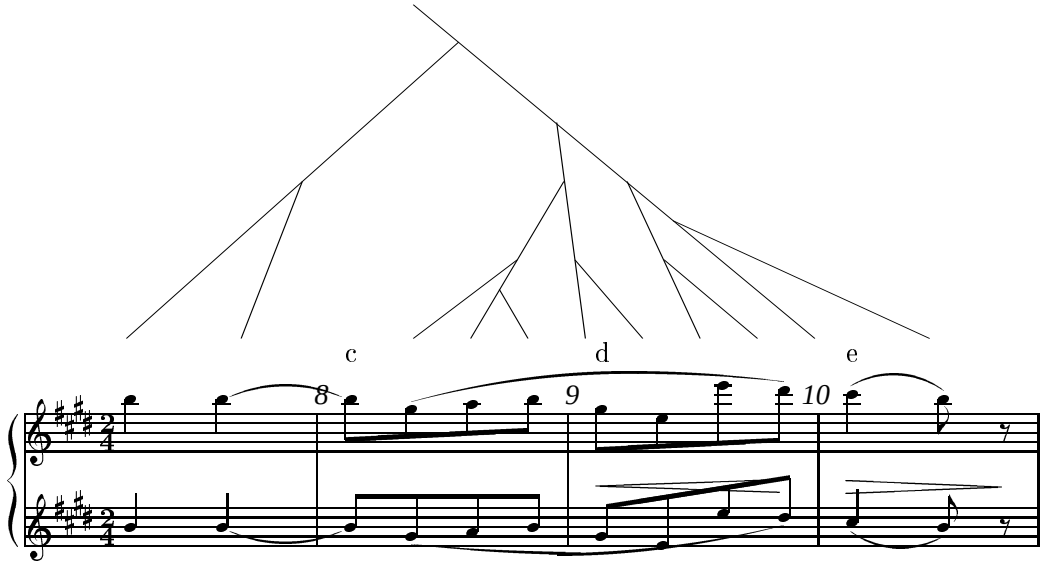


Figure 5: Time-span reduction for bars 7–10

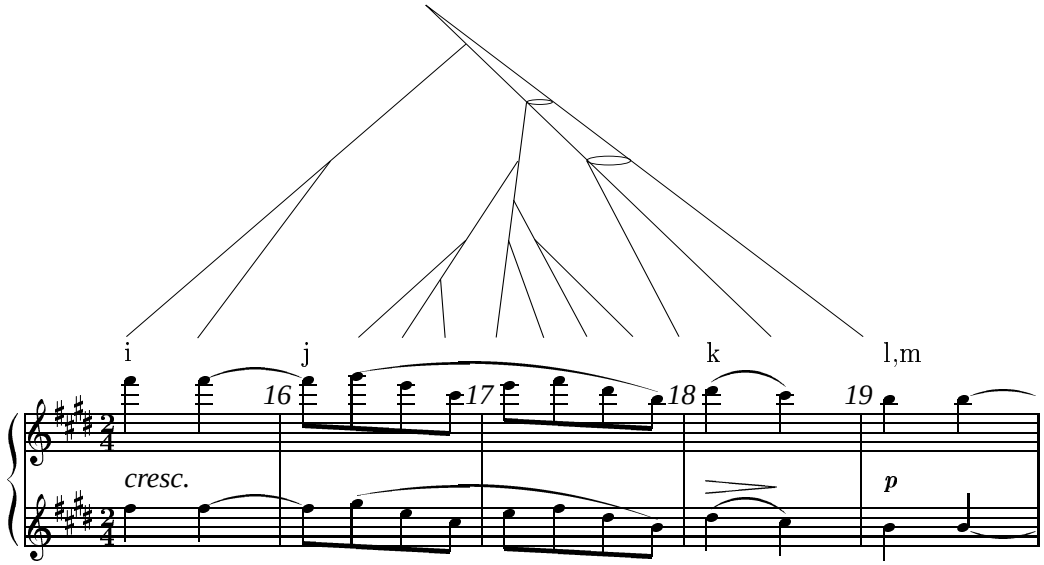


Figure 6: Time-span reduction for bars 15–19

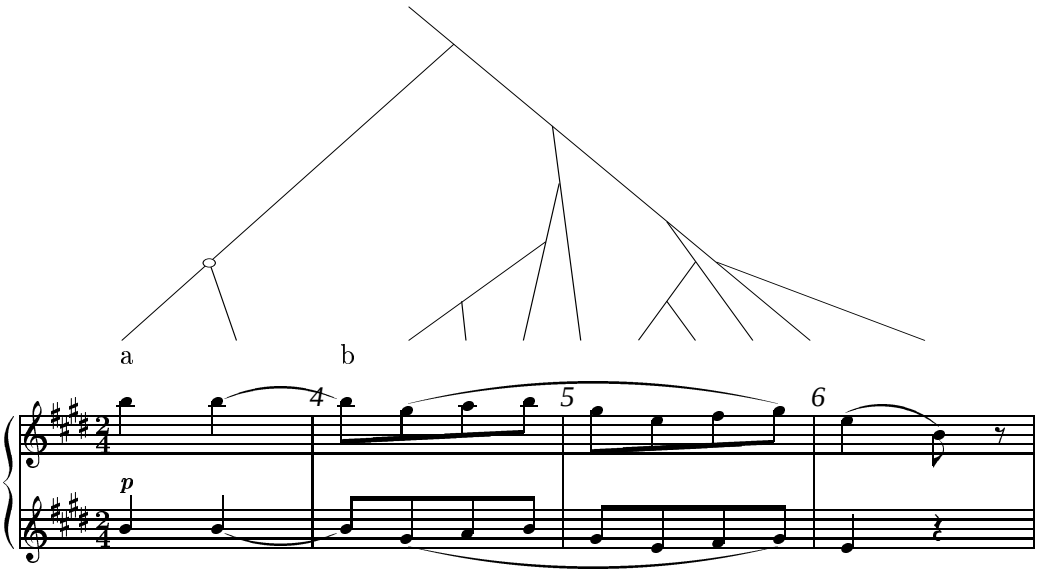


Figure 7: Prolongational reduction for bars 3–6

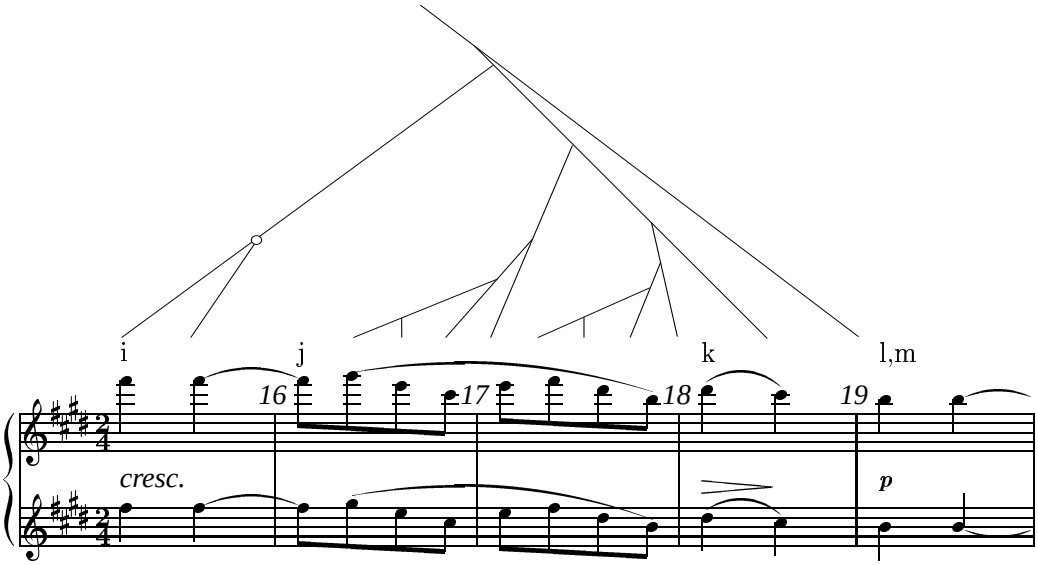


Figure 8: Prolongational reduction for bars 15–19

5.2 Timing Deviations

Before we begin discussing how the structures and timing data interact, we should first point out an interesting, but undesirable, feature of the data collected. Figure 9 shows the timing deviations which took place during all three performances. Although the overall pattern appears to be quite similar, when we inspect the graph more closely we can see that there is quite a lot of deviation between the performances. This deviation is not just in magnitude but in direction meaning that in some performances the same notes were delayed and in others they were played early. This most likely due to the small data set we used and the fact that the piece was not extensively rehearsed by either of the pianists. However, inspection shows that the data is at least usable for a preliminary experiment such as this and we avoid drawing strong conclusions from those parts of the data which are contradictory.

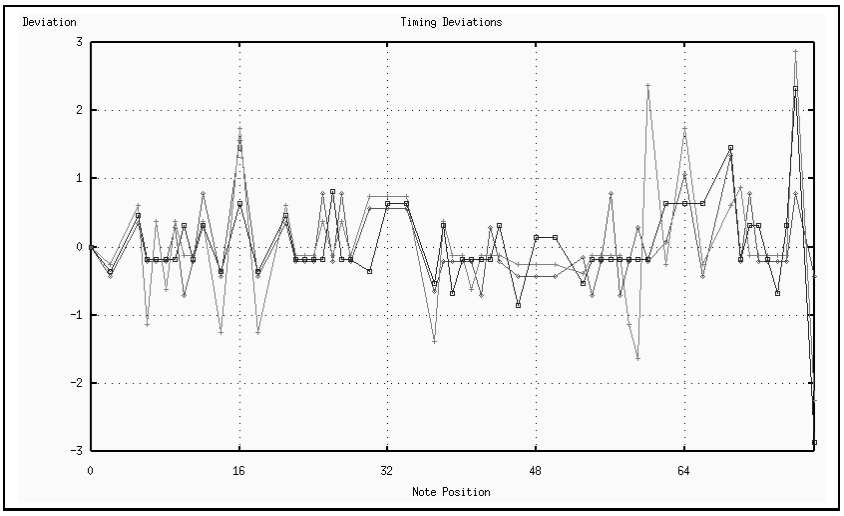


Figure 9: Timing Deviations for all three performances (based upon the duration of the first 22 bars)

5.2.1 Bars 3–10

The score and timing information for bars 3–10 are shown in Figures 10 and 11. We can see that the tie across the barline (marked by b) causes the third note of the phrase to be played significantly earlier than intended. There are then some smaller deviations fluctuating around the melodic rises in bars 4 and 6. The E in bar 6 receives emphasis by the pianist playing it early and then the B is slowed to compensate for this variation. The significance of the E is highlighted both by the time-span analysis and by the prolongational analysis given in Figures 4 and 7.

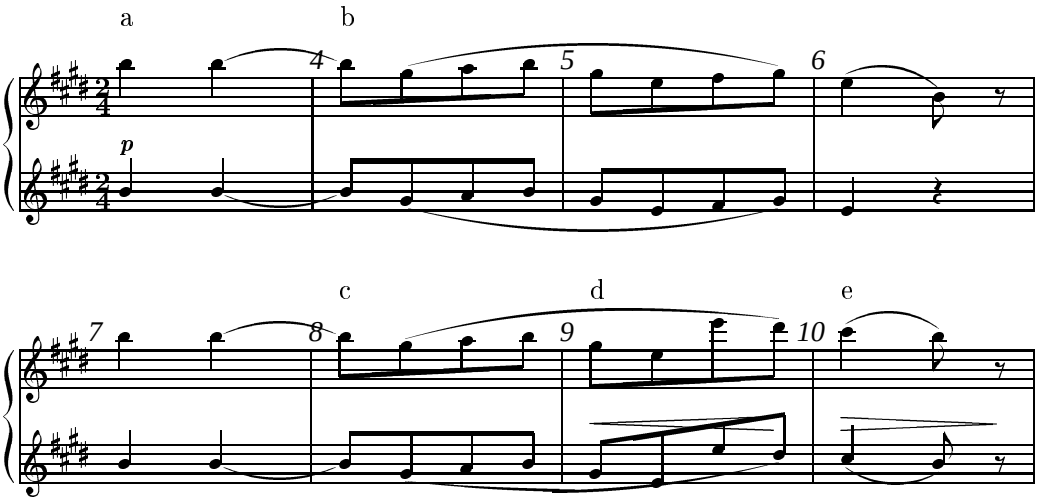


Figure 10: Prima part for bars 3 – 10

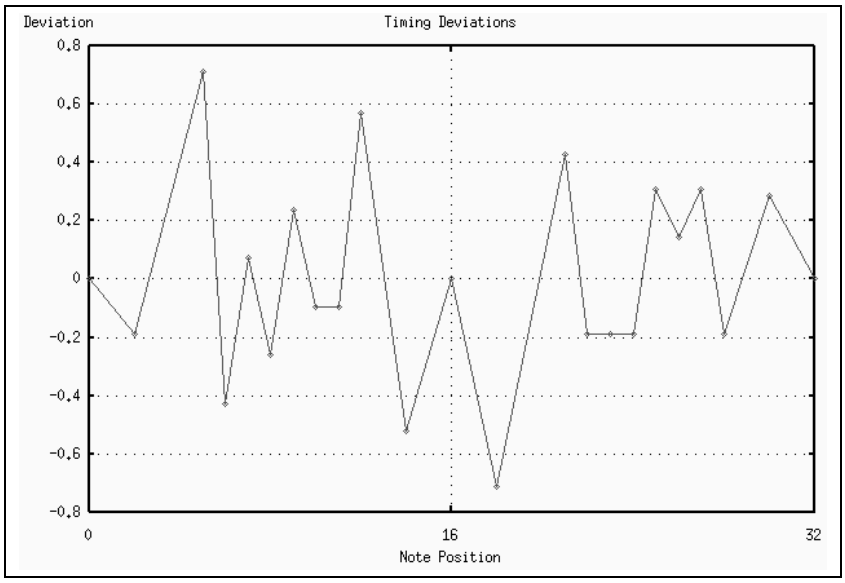


Figure 11: Timing Deviations for bars 3–10

Bars 7 and 8 show a similar pattern of timing deviations leading up to and including the tie (marked by c). However, the second half of the phrase shows a different timing profile to the first. This is probably due to a number of things; the extra grouping boundary caused by the start of the modulation (see Figure 3), the crescendo and the diminuendo (d & e). The results of these differences are clearly visible when the time-span structure in Figure 4 is compared with Figure 5.

5.2.2 Bars 11–18

The first feature of interest in the timing deviation in Figure 13 is the fact that the timing deviations happen in mostly the opposite direction to the deviations in the first phrase. Our tentative explanation for this surprising feature is that the performer is emphasising the results of the modulation and is asserting, particularly with the second event, that we are now in the dominant. This deviation then affects the rest of the phrase.

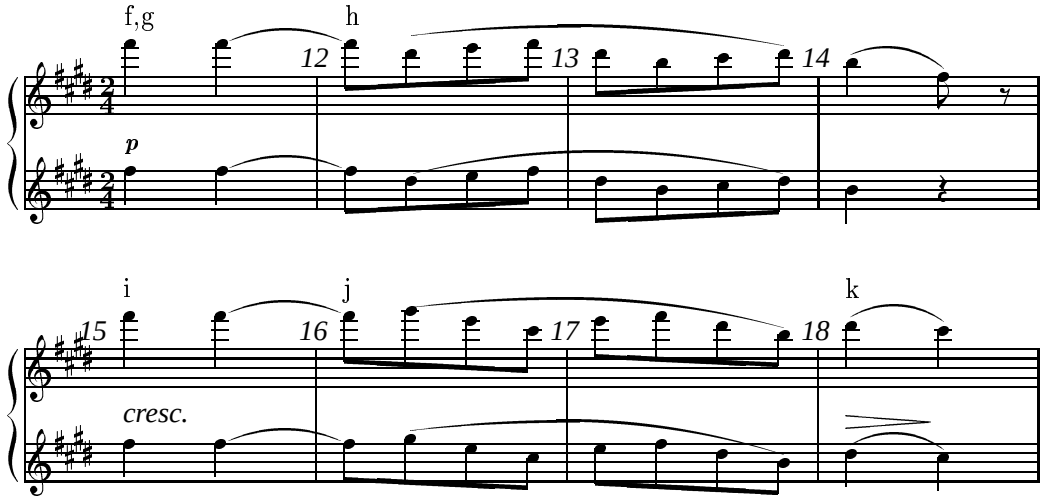


Figure 12: Prima part for bars 11 – 18

The start of bar 15 marks the beginning of a crescendo that leads towards the cadence at the end the phrase. We also have a reversal of the slope in the centre of the phrase so that the melodic rises have become melodic falls. If we look at the time-span reduction for this part (Figure 6) we can see that the middle two bars are viewed as elaborations of the cadence. The events in the middle bars gradually slow down until, when we reach the start of the cadence, the speed then returns to “normal” to help emphasise the cadence. Thus, again, the performance may be said to correspond with the GTTM tree.

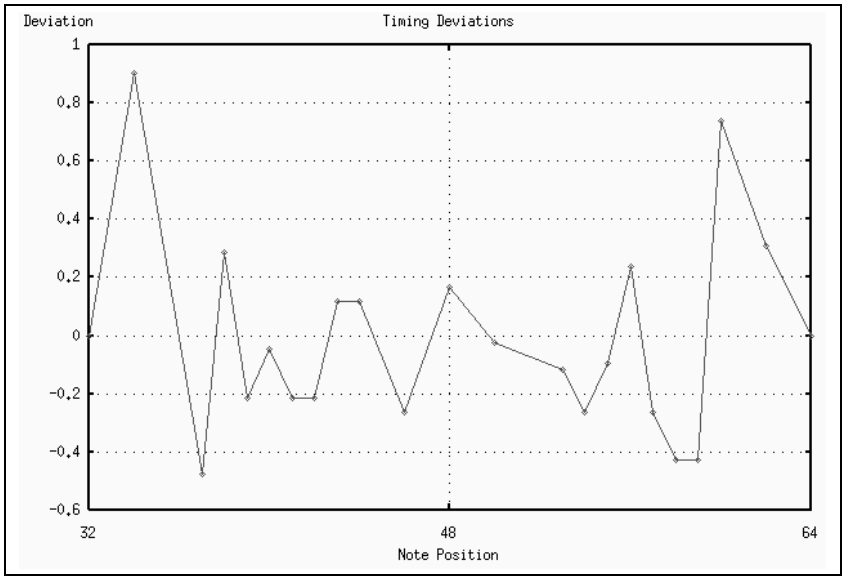


Figure 13: Timing Deviations for bars 11–18

5.2.3 Bars 19–22

The phrase in this section of the piece is an exact copy of the first phrase. We can see the familiar delay of the second note and then the early performance of the third note after the tie (shown as n in Figure 14). The third bar of the phrase continuously slows down until we reach the fourth bar at which point the first note is played very early. Finally, we have a large delay near the end of the phrase which was probably due to the fact that the page needs to be turned at this point. Despite the fact that the first event of this phrase forms part of the cadence of the previous phrase, the structure of this phrase is unaltered as we view the event as being shared between the two phrases.

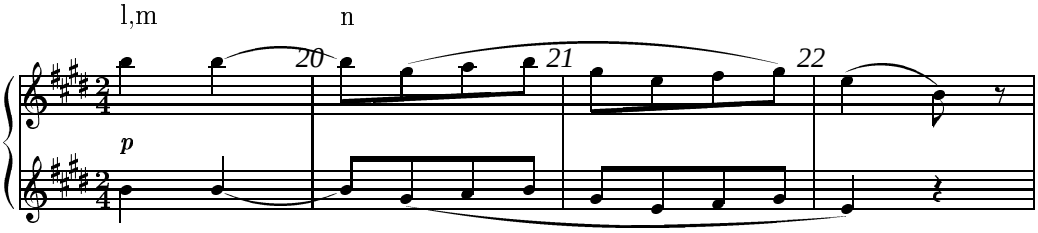


Figure 14: Prima part for bars 19 – 22

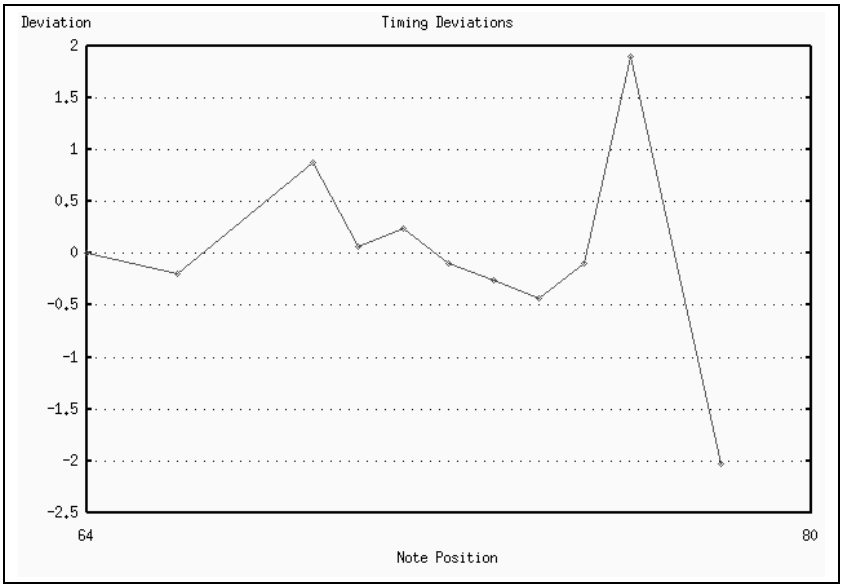


Figure 15: Timing Deviations for bars 19–22

6 Conclusions

The aim of this preliminary experiment was to investigate the relationship between the structure of a piece of music and an expressive performance of that piece. The results suggest that there is indeed a relationship there to be explored. However, looking at our small sample, we can see that there is also quite a lot of "expression" that is not directly accounted for in the structure. Perhaps this is a feature of the data set used, or perhaps we will need to model some aspects of the performers (Parncutt 1997) as well as the piece.

The results of this preliminary experiment do encourage further work to be done in this area. However, when it comes to trying to quantify the relationship between the performance and the structure it will be important to have a larger sample of data to work from. With a large data set, we expect to be able to identify which aspects of the performance are consistent in the majority of the performances and which aspects cannot be accounted for by our research.

Acknowledgements

Ben Curry is supported by UK EPSRC postgraduate studentship 97305827.

References

Arcos, J. L., de Mántaras, R. L., and Serra, X. (1997). Saxex: a case-based reasoning system for generating expressive musical performances. In Cook, P. R., editor, *Proceedings of the International Computer Music Conference*, pages 329–336. Computer Music Association.

Cambouropoulos, E. (1998). *Towards a General Theory of Musical Structure*. PhD thesis, Faculty of Music, University of Edinburgh.

Dannenberg, R. B. (1993). Music understanding by computer. In *IAKTA/LIST International Workshop on Knowledge Technology in the Arts Proceedings*, pages 41–56.

Dannenberg, R. B. and Mukaino, H. (1988). New techniques for enhanced quality of computer accompaniment. In *Proceedings of the International Computer Music Conference*, pages 243–249.

Desain, P., Honing, H., and Heijink, H. (1997). Robust score-performance matching: Taking advantage of structural information. In Cook, P. R., editor, *Proceedings of the International Computer Music Conference*, pages 337–340. Computer Music Association.

Fauré, G. (1894). *Dolly Pour Quatre Mains (op. 56)*. J. Hamelle (Editor), Boulevard Malesherbes, Paris.

Friberg, A., Frydén, L., and Sundberg, J. (1997). A rule for automatic musical punctuation of melodies. In Behne, K.-E., Deliège, I., Gabrielsson, A., and Sloboda, J., editors, *Proceedings of the European Society for the Cognitive sciences Of Music Conference*, pages 719–723. ESCOM.

Grubb, L. and Dannenberg, R. B. (1997). A stochastic method of tracking a vocal performer. In Cook, P. R., editor, *Proceedings of the International Computer Music Conference*, pages 301–308. Computer Music Association.

Lerdahl, F. and Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. MIT Press.

Narmour, E. (1992). *The analysis and cognition of melodic complexity: the implication-realization model*. University of Chicago Press.

Parncutt, R. (1997). Modeling piano performance: Physics and cognition of a virtual pianist. In Cook, P. R., editor, *Proceedings of the International Computer Music Conference*, pages 15–18. Computer Music Association.

Watkinson, S. P. (1997). Induction of musical syntax. Master's thesis, Department of Artificial Intelligence, University of Edinburgh.

Widmer, G. (1995). Modeling the rational basis of musical expression. *Computer Music Journal*, 19(2):76–96.

Representing Trees with Constraints

Ben Curry[1⋆], Geraint A. Wiggins[2] and Gillian Hayes[1]

[1] Institute of Perception, Action and Behaviour, Division of Informatics,
University of Edinburgh, Edinburgh EH1 1HN
[2] Department of Computing, School of Informatics,
City University, Northampton Square, London EC1V 0HB

Abstract. This paper presents a method for representing trees using constraint logic programming over finite domains. We describe a class of trees that is of particular interest to us and how we can represent the set of trees belonging to that class using constraints. The method enables the specification of a set of trees without having to generate all of the members of the set. This allows us to reason about sets of trees that would normally be too large to use. We present this research in the context of a system to generate expressive musical performances and, in particular, how this method can be used to represent musical structure.

1 Introduction

This paper describes how constraints can be used to represent a specific class of trees that have the following properties:

Rooted - each tree has a node distinguished as the root node.
Ordered - the children of each node are distinct and cannot be re-ordered without changing what the tree represents.
Constant depth - the leaf nodes of each tree are all the same distance from the root.
Strict - at each depth, one of the nodes has at least two successors.

The number of distinct trees in this class is large for each n, where n is the number of leaf nodes. If $n \geq 10$ the set of trees described can not easily be manipulated or used within a computer system. We present here an efficient way of representing this large set of trees, using constraint logic programming, that enables us to use this class of trees in our research.

The structure of the paper is as follows. The next section explains why we are interested in representing sets of trees in the context of music. We then present some implementation details including our representation and the constraints used to specify the trees of interest. Some results are presented that illustrate the effectiveness of this method. Finally, we end with our conclusions.

⋆ Ben Curry is supported by UK EPSRC postgraduate studentship 97305827

2 Motivation: Grouping Structure

This work forms part of our research into creating an expressive musical performer that is capable of performing a piece of music alongside a human musician in an expressive manner.

An expressive performance is one in which the performer introduces variations in the timing and dynamics of the piece in order to emphasise certain aspects of it. Our hypothesis is that there is a direct correlation between these expressive gestures and the musical structure of the piece and we can use this link to generate expressive performances.

The theory of musical structure we are using is the Generative Theory of Tonal Music (GTTM) by Lerdahl and Jackendoff (1983). The theory is divided into four sections that deal with different aspects of the piece's musical structure. We are particularly interested in the *grouping structure* which corresponds with how we segment a piece of music, as we are listening to it, into a hierarchy of groups. It is this hierarchy of groups that we seek to represent with our trees.

The rules are divided into two types: *well-formedness* rules that specify what structures are possible; and *preference* rules that select, from the set of all possible structures, those that correspond most closely to the score.

The rules defining grouping structures are based on principles of change and difference. Figure 1 shows four places where a grouping boundary may be detected (denoted by a '$*$'). The first case is due to a relatively large leap in pitch between the third and fourth notes in comparison to the pitch leaps between the other notes. The second boundary occurs because there is a change in dynamics from piano to forte. The third and fourth boundaries are due to changes in articulation and duration respectively.

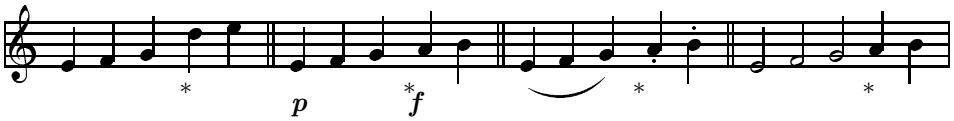


Fig. 1. Points in the score where grouping rules may apply

Figure 2 shows an example of a grouping structure for a small excerpt of music. We can see that the music has been segmented into five different groups, one for each collection of three notes. The musical rest between the third and fourth groups causes a higher level grouping boundary that makes two higher level groups which contain the five groups. These groups are then contained within one large group at the highest level.

The grouping structure can be represented with a tree. Figure 3 shows a tree representation (inverted, to aid comparison) for the grouping structure shown in Fig. 2. The leaf nodes at the top of the tree correspond to the notes in the score, and the branches convey how the notes are grouped together. This is an

example of the class of tree we are trying to represent. From this point onwards the trees will be presented in the more traditional manner, i.e. the leaf nodes at the bottom and the root node at the top.

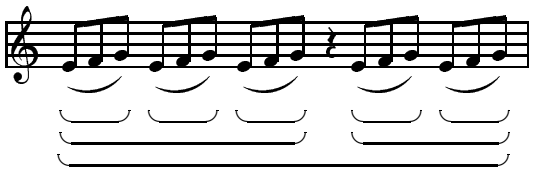


Fig. 2. An example grouping structure

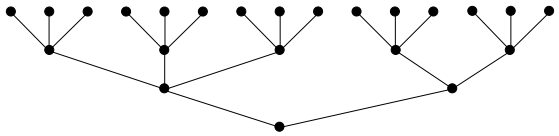


Fig. 3. Tree representing the grouping structure shown in Fig. 2

Although the GTTM grouping rules are presented formally, the preference rules introduce a large amount of ambiguity. For a particular piece of music, there are many possible grouping structures which would satisfy the preference rules. The purpose of the present research is to devise a way to represent this large set of possible structures in an efficient way so that they can be used by a computer system.

Using our hypothesis of the link between musical structure and expressive performance, one of the core ideas of our research is to use rehearsal performances by the human musician to disambiguate the large set of possible grouping trees. The expressive timing used by the musician in these rehearsals provides clues as to how the musician views the structure of the piece. A consistent pattern of timing deviations across a number of performances will enable us to highlight points in the score where the musician agrees with the possible grouping boundaries.

3 Using Constraints

This section of the paper explains how we use constraint logic programming (Van Hentenryck, 1989) to represent sets of trees. Although constraints have been used in the areas of music composition (e.g. Henz 1996) and tree drawing

(e.g. Tsuchida 1997), this research is concerned with an efficient representation of large numbers of tree structures, which is a problem distinct from these.

Constraint logic programming over finite domains enables the specification of a problem in terms of variables with a range of possible values (known as the *domain* of the variable) and equations that specify the relationships between the variables. For example if (1), (2) and (3) hold then we can narrow the domains of x and y as shown in (4):

$$x \in \{1..4\} \tag{1}$$
$$y \in \{3..6\} \tag{2}$$
$$x + y \geq 9 \tag{3}$$
$$x \in \{3..4\} \wedge y \in \{5..6\} \tag{4}$$

The following sections outline the representation and the constraints we use to specify the class of trees. We begin by discussing the representation of the nodes and then present the five types of constraints used to ensure that the trees generated belong to our class.

3.1 Representation

We know that our class of trees will be monotonically decreasing in width from the leaf nodes up to the root and, therefore, we can represent the set of trees by a triangular point lattice of nodes[1]. Figure 4 shows the point lattices for trees of width $n = 3$ and $n = 4$.

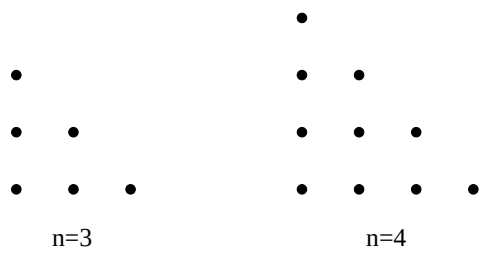


Fig. 4. Point lattices for trees of width 3 and 4

Each node has the following variables (illustrated in Fig. 5):

1. *id*: a unique identifier;
2. *uplink*: a connection to the level above;

[1] An implementation detail means that there is always a path from the highest node of the point lattice to the leaf nodes, but this highest node should not be considered the root node. The root node may occur at any height in the point lattice and is identified as the highest node with more than one child.

3. Downlink values which represent all the nodes on the level below that are connected to this one.

The *id* is specified as an (x,y) coordinate to simplify the implementation details. The *uplink* variable contains an integer that represents the x-coordinate of the node on the level above to which this node is connected i.e. node ($uplink$, $y + 1$). The downlink values, specified by a lower (dl) and upper (du) bound, refer to a continuous range of nodes on the level below that may be connected to this one i.e. nodes (dl, $y - 1$)...(du, $y - 1$).

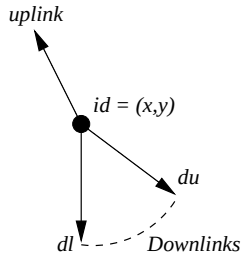


Fig. 5. A typical node

The next sections present the constraints that are applied to the nodes in order to create the specific set of trees in which we are interested. They begin by specifying the domains of the variables and then constraining the nodes so that only those trees that belong to our class can be generated.

3.2 Node Constraints

The first task is to define the domains of the variables for each node. Due to the triangular shape of the point lattice, the uplink for each node is constrained to point either upwards, or up and to the left of the current node. We constrain the downlink for each node to span the nodes directly below, and below and to the right of the current node.

The constraints (given in (5)-(8)) define the domains of the *uplink* and downlink range (i.e. dl and du) for each node[2]. The *uplink* lies in the range $\{0..x\}$ where x is the x-coordinate of the current node. The zero in the range is used when the node is not connected to the level above.

$$domain([uplink]) = \{0..x\} \tag{5}$$
$$domain([dl, du]) = \{0..n\} \tag{6}$$
$$(dl = 0) \oplus (dl \geq x) \tag{7}$$
$$du \geq dl \tag{8}$$

[2] The $\oplus$ in (7) denotes exclusive-or.

The downlink specifiers dl and du are constrained in a similar way to lie in a range from {0..n} with the added constraints that du has to be greater than or equal to dl and that dl either equals zero or is greater than or equal to x. Figure 6 shows how these constraints relate to the direction of the connections to and from each node.

Constraint (9) handles the situation of a node which is not used in a tree. If the *uplink* of the node is zero then the downlinks of the node must also be zero.

$$((dl = 0) \Leftrightarrow (du = 0)) \wedge ((dl = 0) \Leftrightarrow (uplink = 0)) \tag{9}$$

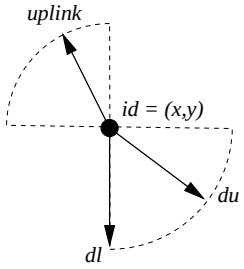


Fig. 6. Constraining the Uplinks and Downlinks

3.3 Level Constraints

To ensure that the connections between two levels do not cross, constraints (10) and (11) are applied to each pair of adjacent nodes. For a pair of nodes A and B, with A directly to the left of B, the $uplink_B$ must either point to the same node as the $uplink_A$ or to the node to the right of it or, if it is unused, be equal to zero (10).

$$(uplink_B = uplink_A) \vee (uplink_B = uplink_A + 1) \vee (uplink_B = 0) \tag{10}$$

Once one of the uplinks on a particular level becomes equal to zero, all the uplinks to the right of it must also be zero (11). This prevents the situation of an unconnected node in the midst of connected ones.

$$(uplink_A = 0) \Rightarrow (uplink_B = 0) \tag{11}$$

Figure 7 shows examples of correct and incorrect mid-sections of a tree under these new constraints. The bottom example is incorrect because it violates constraints (10) and (11).

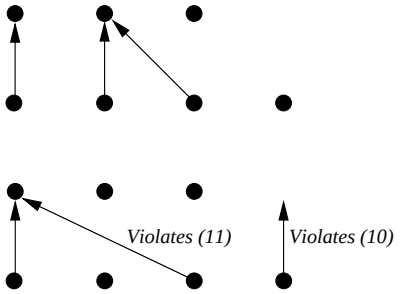


Fig. 7. A correct (*top*) and incorrect (*bottom*) mid-section of a tree

3.4 Consistency Constraints

If the current node refers to a node in the level above, the x-coordinate of this node must appear within its downlink range. Constraint (12) ensures that if this node points to a node on the level above, the downlink range of that node must include this one. Figure 8 shows how this constraint affects two nodes where the lower one is connected to the upper one.

$$(x_{above} = uplink_{this}) \Leftrightarrow ((x_{this} \geq dl_{above}) \wedge (x_{this} \leq du_{above})) \qquad (12)$$

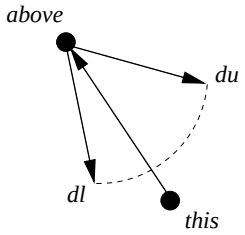


Fig. 8. Ensuring connectivity between nodes

3.5 Width Constraints

We now constrain the trees to decrease in width as we travel from the leaf nodes to the root node. The width of a level is defined as the number of nodes that have a non-zero uplink on that level. Constraint (13) deals with this situation with the precondition that the width of the current level is greater than 1. This precondition is necessary to allow situations such as the first four trees in Fig. 10 where we consider the root node to be at the point where branching begins.

$$(width_i > 1) \Rightarrow (width_j < width_i) \tag{13}$$

We want to ensure that the trees decrease in width to reduce the search space as much as possible. Figure 9 shows an example of a tree which does not decrease in width between two levels, we can remove this tree from our search space as it does not contribute anything new to the grouping structure as we move from level i to level j.

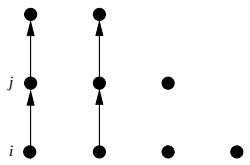


Fig. 9. A section of a tree that does not decrease in width

3.6 Edge Constraints

The last step is to ensure that the uplink of the rightmost[3] node on a level points inwards (the rightmost node in Fig. 7 is an example of this). We find the maximum x of the level above that has a non-zero *uplink* and then ensure that the *uplink* of the rightmost node points to it ((14) and (15)).

$$\mathcal{S} = \{x : id(x, y) \text{ has } uplink_x \neq 0\} \tag{14}$$

$$uplink \leq max(\mathcal{S}) \tag{15}$$

3.7 Valid Trees

The constraints given in §3.2 to §3.6 define the set of trees which belong to our class. Figure 10 shows an example set of width $n = 4$. The white nodes are ones that appear in the generated solutions but are not considered to be part of the tree since the root of the tree is the highest node with more than one child.

3.8 Using the Constraint Representation

The constraints which have been defined in the sections above describe a general class of trees. The next step is to introduce aspects of the grouping structure to

[3] By 'rightmost' we mean the node on the current level with the maximum x-coordinate that has a non-zero uplink.

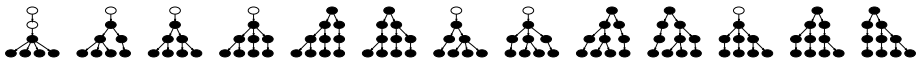


Fig. 10. All the trees of width four ($n = 4$)

reduce this large set of trees to only those trees that correspond to the piece of music being analysed.

Every point in the musical score where a grouping boundary could occur is identified, for each of these points we then measure the relative strength of this boundary against the surrounding ones. Every boundary point can then be used to determine the shape of the tree by ensuring that every pair of notes intersected by a boundary corresponds to a pair of nodes separated in the tree set.

To separate the nodes in a tree, we need to ensure that the parents of the nodes are not the same, and if we have a measure of relative strength between boundaries, we can specify how far towards the root the nodes need to be separated. The algorithm below shows how this is implemented:

$Repel(id_A, id_B, strength)$
  if $(strength \geq 1)$ **then**
    $parent(id_A) \neq parent(id_B)$
    $Repel(parent(id_A), parent(id_B), strength - 1)$
  endif

This recursive predicate takes two nodes and a strength argument and recursively ensures that the nodes are separated up to a height *strength*. Figure 11 shows an example tree where the tree is divided into two subtrees by a *Repel* constraint that is applied with $strength = 1$ between the second and third leaf nodes.

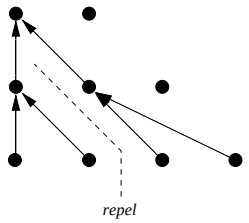


Fig. 11. How *Repel* affects the tree

4 Results

We generated all the trees up to width $n = 7$ and found a similarity with an entry in the Online Encyclopedia of Integer Sequences (Sloane, 2000). It matched a sequence discovered by the mathematician Arthur Cayley (1891) based upon this particular class of trees which has the recurrence shown in (16) and (17)[4]

This recurrence defines the number of trees that belong to our class that are of width n.

$$a(0) = 1 \tag{16}$$

$$a(n) = \sum_{k=1}^{n} \binom{n}{k} a(n-k) \tag{17}$$

Using our representation, the approximate formula, derived experimentally, for the number of constraints to represent the set of all the trees of width n is given in (18).

$$Constraints \approx \frac{2}{3}n^3 + 11n^2 - \frac{2}{3}n - 24 \tag{18}$$

The number of trees of width n grows rapidly (e.g. the number of trees of width 50 is 1.995×10^{72}). By contrast, the number of constraints it takes to represent the same number of trees is 1.1×10^5.

Figure 12 shows how the number of trees grows in comparison to the number of constraints as we increase the width of the tree. The number of trees increases at a greater than exponential rate whereas the number of constraints increases at a low-order polynomial rate.

5 Conclusions

This paper presents our research on representing a specific class of trees with constraint logic programming. Although the number of constraints needed to represent these large sets of trees is comparatively small, the computational time needed to solve the constraints is not.

The representation currently restricts the trees to have leaf nodes at the same depth; however, it does allow the addition of quite simple constraints to change the class of trees represented. For example, to restrict the trees to strictly binary trees we need only add the constraint $du = dl + 1$.

With the use of constraints we have delayed the generation of trees until we have added all the possible restrictions, this offers a great reduction in complexity and allows us to manipulate trees of greater width than would normally be possible.

[4] Where $\binom{n}{k}$ is the standard n choose k formula given by: $\frac{n!}{k!(n-k)!}$

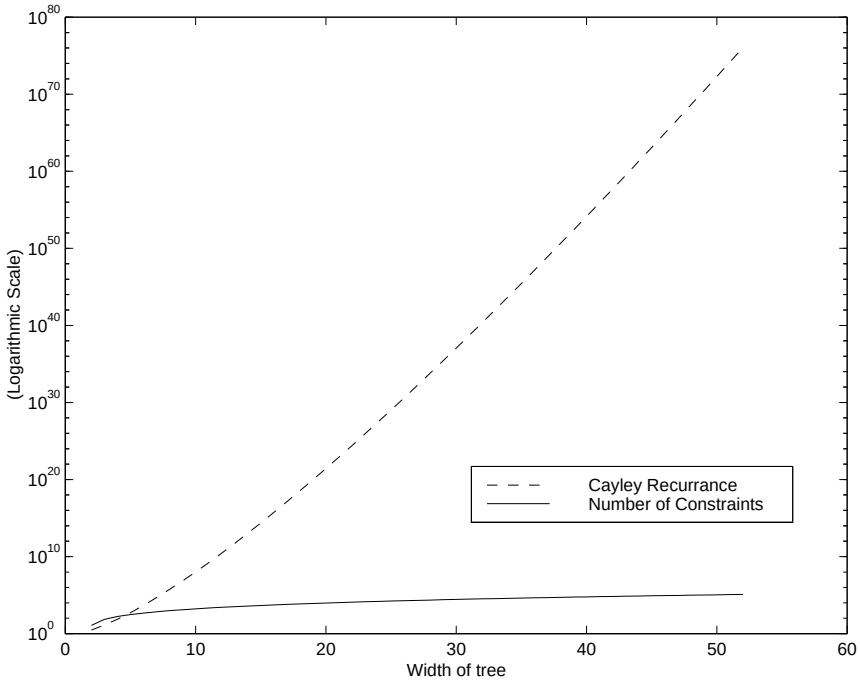


Fig. 12. A graph showing how the number of trees and number of constraints grows with the width of the tree

References

Cayley A.: On the Analytical Forms Called Trees. Coll. Math. Papers, Vol. 4. Cambridge University Press (1891)

Henz M., Lauer S. and Zimmermann D.: COMPOzE – Intention-based Music Composition through Constraint Programming. Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence, IEEE Computer Society Press (1996)

Lerdahl F. and Jackendoff R.: A Generative Theory of Tonal Music. MIT Press (1983)

Sloane N. J. A.: The On-Line Encyclopedia of Integer Sequences. Published electronically at http://www.research.att.com/~njas/sequences/ (2000)

Tsuchida K., Adachi Y., Imaki T. and Yaku T.: Tree Drawing Using Constraint Logic Programming. Proceedings of the 14th International Conference of Logic Programming, MIT Press (1997)

Van Hentenryck P.: Constraint Satisfaction in Logic Programming. Logic Programming Series, MIT Press (1989)

Appendix E

Musical Scores

E.1 *Berceuse*

E.2 *Auf dem Hügel sitz ich spähend*

E.3 *Gute Nacht*